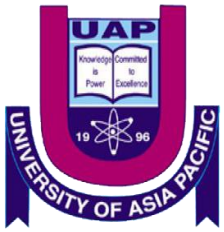


University of Asia Pacific

Department of CSE

Group Id: A2 – G3

Project Name: Disease Prediction System Web App

Project directory link: GitHub Link

Project Team Leader: Apurba Datta (20101040)

Project members:

- Apurba Datta (20101040)
- Mahir Tajwar Bhuiyan (20101034)
- Md. Fayaj Nakib(20101036)
- Sanjana Afrin(20101033)

Motivation

Developing a Disease Prediction System project aims to improve healthcare outcomes by detecting diseases early and providing personalized treatment recommendations. This can be achieved by analyzing user input data and medical datasets to identify patterns that may indicate the presence of a disease or the risk of developing one.

The Disease Prediction System using machine learning has several motivations, including:

1. Improving healthcare: The disease prediction system can help predict diseases early, and healthcare providers can take necessary steps to prevent or manage the disease before it progresses to a more severe stage, leading to better healthcare outcomes
2. Saving time and resources: Automated disease prediction can help healthcare providers to save time and resources by reducing the need for manual testing and diagnosis.
3. Cost efficiency: Additionally, the project has the potential to reduce healthcare costs by improving the efficiency of disease detection and treatment.

Existing and similar systems:

Meisam Shabanpoor, and Mehregan Mahdavi in "Application of recommender systems on disease recognition and treatment" suggested an idea to develop an application for disease recommender systems [1]. "Multiple Disease Prediction Webapp", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and ISSN Approved), suggested an idea to develop an application for disease recommender systems using streamlit framework [2]. Md. Tahmid Rahman Laskar, Md. Tahmid Hossain, Abu Raihan Mostofa Kamal, and Nafiul Rashid in "Automated Disease Prediction System (ADPS): A User Input-based Reliable Architecture for Disease Prediction" explained the proper working of the automated disease prediction system and the working of the algorithm [3]. Kedar Pingale, Sushant Surwase, Vaibhav Kulkarni, Saurabh Sarage, and Prof. Abhijeet Karve in "Disease Prediction Using Machine Learning" discussed various algorithms which can be implemented for the disease prediction model. This system is used to predict most chronic diseases. It accepts the structured and textual type of data as input to the machine learning model [4]. We

have utilized a supervised learning methodology for building the prediction model [5]. We have chosen classification algorithms (logistic regression or SVM) to build our model because they suit best disease prediction models [6].

Inspired by those systems, we believe that there is potential to achieve more from those systems. So, we reconstruct a similar system but use a different machine learning model to get a more accurate result.

Problem statement

The primary goal is to create a prediction engine that enables users to check for the presence of diabetes or heart disease from their homes, without needing to visit a doctor for further treatment. The prediction engine will require a large dataset and efficient machine learning algorithms to accurately predict the likelihood of the disease. To achieve optimal performance of the prediction engine, the dataset needs to be pre-processed by removing redundant, null, or invalid data and training the machine learning models.

Doctors often rely on their experience and past cases to treat patients. But when dealing with new or rare diseases, this can be a slow and uncertain process. Machine learning can help identify patterns earlier and improve treatment outcomes.

This project aims to analyze the given data and train the data using various algorithms and then predict the diseases on the basis of the machine learning model and develop the user interface for the same.

We have utilized a supervised learning methodology for building the prediction model. We have chosen classification algorithms (logistic regression or SVM) to build our model because they suit best disease prediction models.

Objectives

The objective of the project (multiple diseases prediction system web app) is to provide users with an easy-to-use online tool that can help them identify potential health risks and/or diagnose certain medical conditions based on their symptoms or other relevant factors such as age, gender, lifestyle habits, and family history. This data is then used to develop a machine learning or other predictive model that can analyze the input data and generate personalized health recommendations.

One of the essential aspects of this project is building an intuitive and user-friendly interface that allows users to input their information and receive results in real time. Additionally, robust data security features are incorporated to ensure the privacy and confidentiality of user information.

The accuracy and reliability of the predictive model are tested and validated using real-world data sets. Furthermore, the model is continually updated and incorporates new research findings and data sources to improve the accuracy of predictions.

The goal of a multiple diseases prediction system web app project is to provide individuals with actionable insights and treatment plans based on their risk factors and predicted outcomes. Ultimately, this tool empowers individuals to take charge of their health and make informed decisions about their well-being.

Project output

The output of such a project would typically involve a user interface built using the Streamlit framework. This interface would allow users to input relevant data and would display the predicted diseases or health conditions based on the input data.

To develop such a system, the following steps could be followed:

1. **User-friendly interface:** The web app should have a user-friendly interface that allows users to easily navigate and use the app. It should be easy to use and understand, even for those without a technical background.
2. **Disease prediction:** The web app should provide accurate predictions for multiple diseases based on the user's input. The user should be able to input their symptoms, medical history, and other relevant information to get an accurate prediction.
3. **Fast and efficient**: The web app should provide fast and efficient predictions, with minimal delays or downtime. Users should not have to wait too long for the results.
4. **Privacy and security:** The web app should ensure the privacy and security of the user's information. The user's information should be kept confidential and secure, and should not be shared with third parties without the user's consent.
5. **Customization:** The web app could allow users to customize their experience, such as setting preferences for notifications or receiving alerts for specific conditions.

Overall, the output of the multiple diseases prediction system web app should be an accurate and reliable tool that helps users identify potential health conditions and take appropriate action. The web app should be designed with the user's needs in mind and should provide a user-friendly and efficient experience.

Effect on society

Here are some key points on the potential effects of a disease prediction system using machine learning on society.

1. **Improved Healthcare Outcomes:**
    - Early detection and treatment of diseases can lead to better health outcomes and improved quality of life for patients.
    - Machine learning can help doctors make faster and more accurate diagnoses, leading to better patient health outcomes.

2. **Improved healthcare access:**
    - Healthcare providers can use the app to remotely diagnose and monitor patients, which can be especially helpful in rural areas or for patients who cannot easily access healthcare services.

3. **Cost-effective healthcare:**
    - Machine learning-powered diagnosis and monitoring can help reduce healthcare costs by streamlining the diagnostic process and reducing unnecessary tests and procedures.

4. **Better patient outcomes:**
    - Early detection and prevention can lead to better treatment outcomes and improved quality of life for patients.
5. **Increased public health awareness:**
    - Disease prediction apps can increase public awareness about various diseases and their risk factors, leading to a more informed and health-conscious population.

Requirement analysis

Basic Requirement:

1. **User Interface:** The web app should have a simple and intuitive user interface that allows users to interact with the application easily.
2. **AL-based tool:** The application should have an AL-based machine learning model that can automatically predict the likelihood of a user having a certain disease based on input parameters.
3. **Input Fields:** The web app should have input fields for the user to input their personal and medical information, such as age, gender, blood pressure, and other relevant symptoms or indicators.
4. **Prediction Results:** The app should display the prediction results to the user in a clear and understandable way.
5. **Performance:** The web app should be optimized for performance, ensuring quick and accurate predictions.

Functional Requirement:

1. **Disease Prediction:** The main functionality of the web app is to predict the likelihood of a user having a particular disease based on their input parameters.
2. **Data Preprocessing:** The app should preprocess and clean the user's input data before passing it to the machine learning model for prediction.
3. **Machine Learning Algorithm:** The web app should use an appropriate machine learning algorithm to make accurate predictions based on the user's input data.
4. **Multiple Disease Prediction:** The app should allow the user to predict multiple diseases based on their input parameters.
5. **Model Training:** The machine learning model should be trained on a relevant dataset to ensure accurate predictions.
6. **Model Deployment:** The trained model should be deployed on a web server(streamlit) or cloud platform to enable online predictions.
7. **Scalability:** The web app should be designed to handle a large number of users and prediction requests.

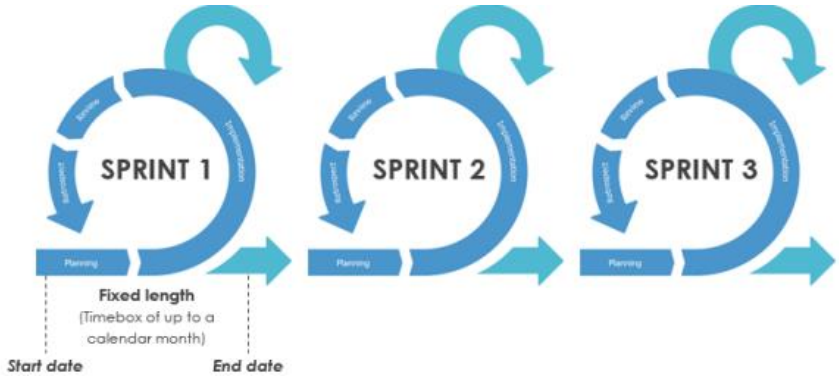
Technical Requirement

Agile software development

The Agile software development process model is a popular approach for developing software products in an iterative and incremental manner. It emphasizes collaboration between cross-functional teams, continuous feedback, and delivering working software at regular intervals.

Here we use one of the most popular Agile design process models for developing a disease prediction system using machine learning the Scrum framework. SCRUM is an agile development method that concentrates specifically on how to manage tasks within a team-based development environment. Scrum encourages teams to learn through experiences and self-organize while working on a problem.

Each iteration of a scrum is known as Sprint. Scrum has three roles product owner, Scrum Master, and Team member.



The Main Artefacts

- **Product Backlog** is the master list of work that needs to get done and maintained by the product owner or product manager.
- **Sprint Backlog** is the list of items, user stories, or bug fixes, selected by the development team for implementation in the current sprint cycle.

Agile Sprint Table:

Sprint	Duration	Sprint Goal	Tasks	Backlog	Steps
1	3 weeks	Set up project and user interface	1. Create a project plan and backlog 2. Implement UI 3. Develop user registration and login functionality	Create a project plan and backlog and Implement symptom checker UI and user authentication	Requirement Analysis, Design, Code, Test, Learning
2	3 weeks	Implement symptom checker and disease prediction, algorithm	1. Develop a disease prediction algorithm 2. Test symptom checker and prediction algorithm	Develop a disease prediction algorithm	Requirement Analysis, Design, Code, Test, Learning
3	2 weeks	Disease prediction information pages	1. Develop disease information pages 2. pages	Develop disease information pages.	Requirement Analysis, Design, Code, Test, Learning
4	3 weeks	Improve user experience and performance	1. Optimize page load times 2. Improve UI and user experience 3. Test performance and user experience improvements	Optimize app performance and Improve UI/UX design	Requirement Analysis, Design, Code, Test, Learning

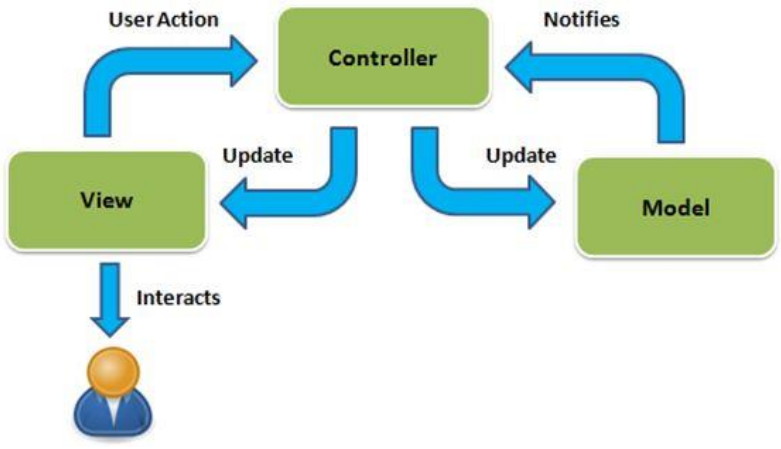
Design pattern:

Here we use an MVC design pattern to design a disease prediction system using machine learning.

1. Model: This component represents the application's data and business logic. It is responsible for managing the data and the operations performed on that data.
2. View: This component represents the user interface of the application. It is responsible for rendering the data from the model and presenting it to the user. 3. Controller: This component acts as an intermediary between the model and the view. It receives input from the user via the view, processes that input through the model, and updates the view with the result.

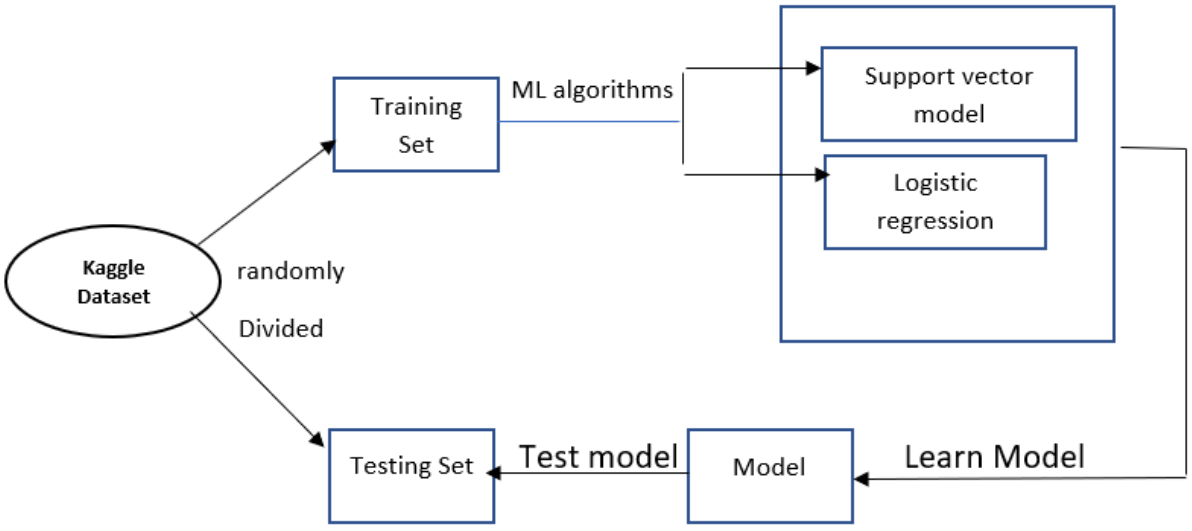
Here is a simplified example of how the MVC components could interact:

1. The user interacts with the view by entering their personal and health-related data.
2. The view passes the user input to the controller, which validates and sanitizes it.
3. The controller passes the sanitized data to the model for processing.
4. The model applies machine learning algorithms to the data and generates a prediction.
5. The model passes the prediction back to the controller.
6. The controller passes the prediction to the view for display to the user.



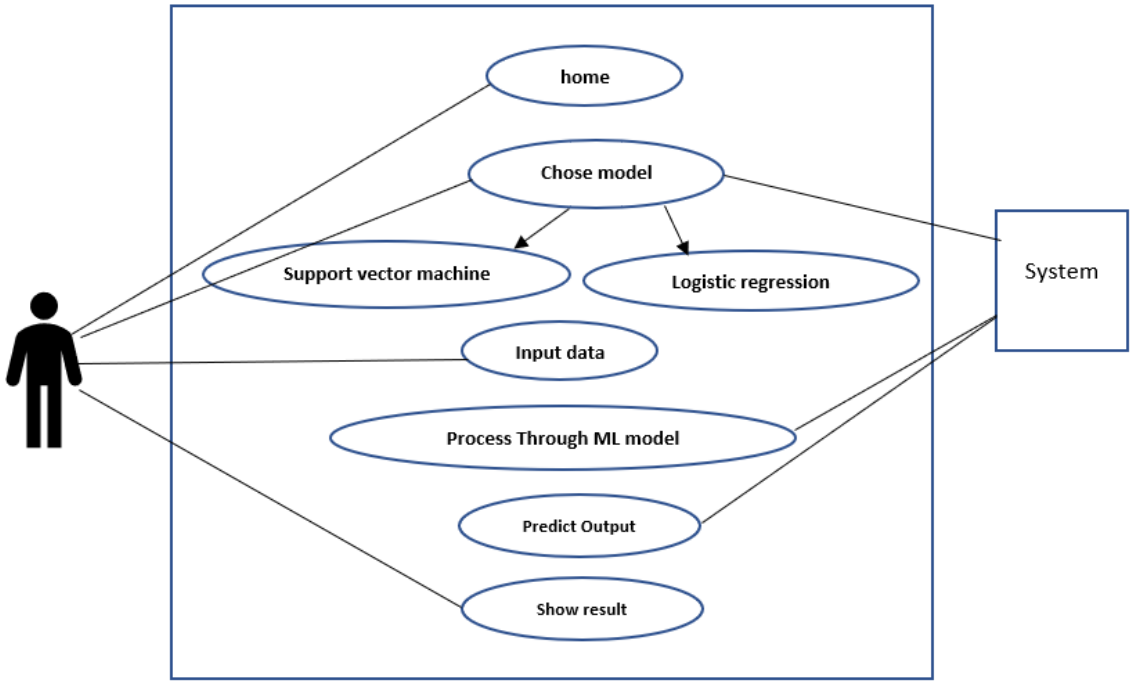
Methodology:

SYSTEM ARCHITECTURE:



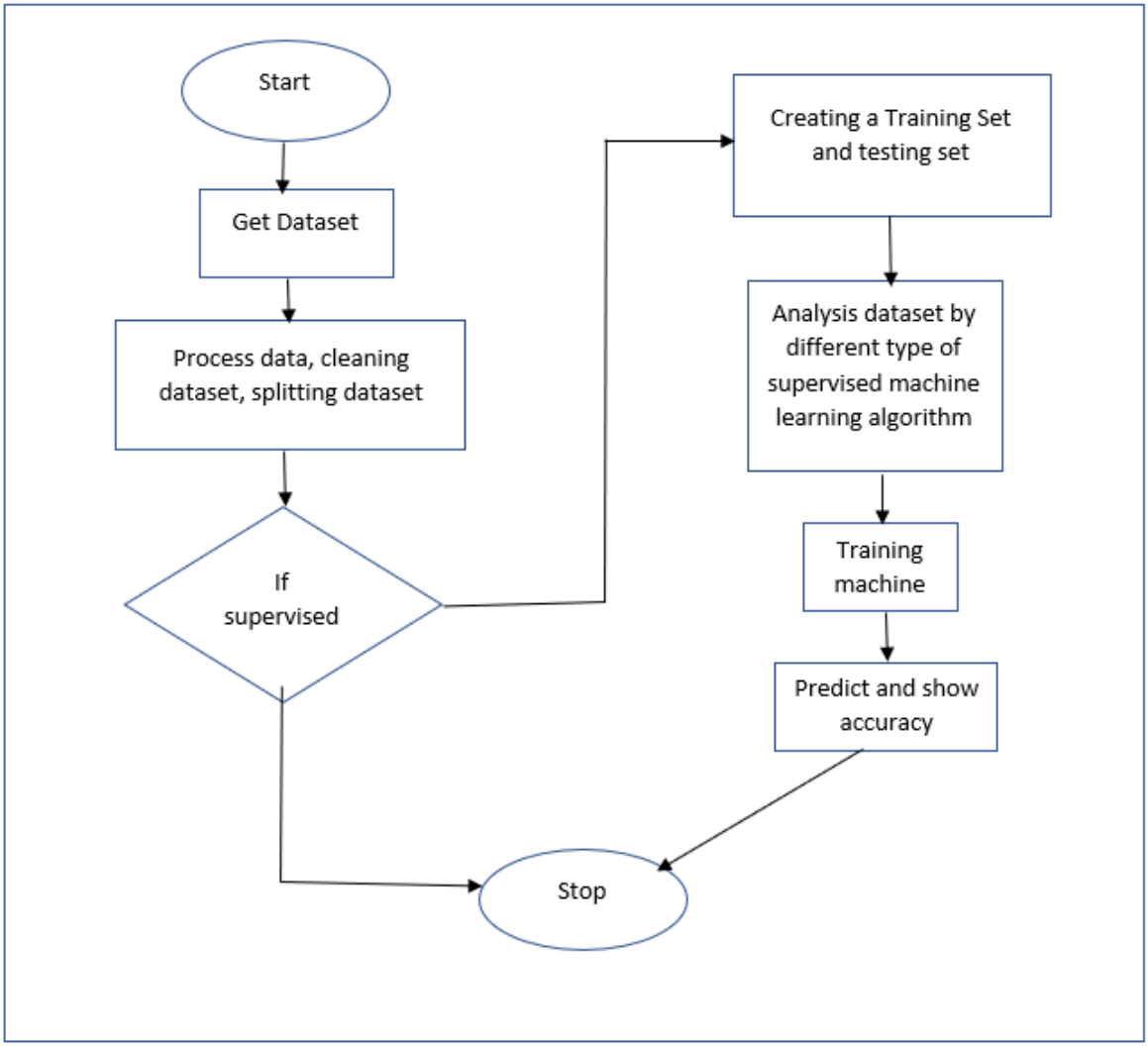
Show different types of diagrams for diseases prediction systems:

UML diagram:



DFD: DATA FLOW DIAGRAM

In the context of the disease prediction web app, a DFD could be created to represent how data flows through the app, including how user inputs are processed, how the disease prediction algorithm is applied, and how the results are presented to the user.



Software Process Model

here's an example of how the Agile software process model with sprints could be used for developing a disease prediction system using machine learning with an SVM model:

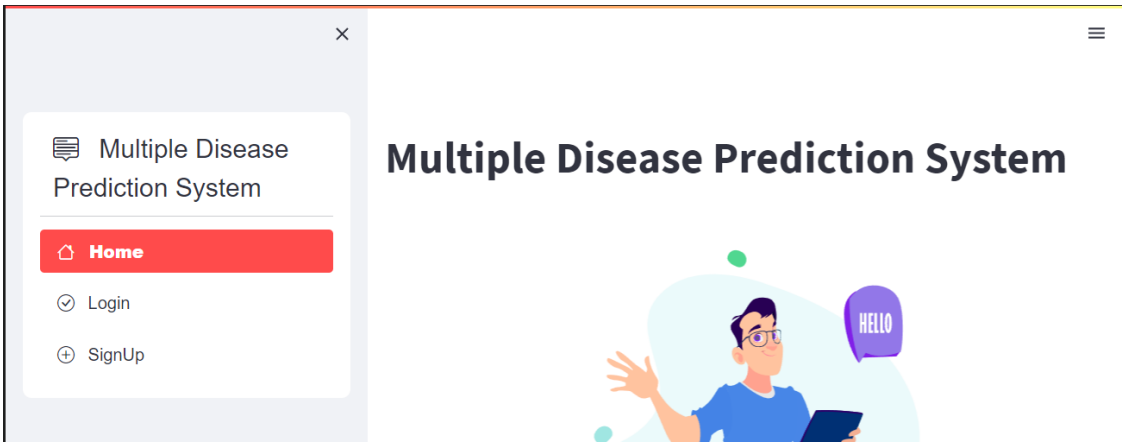
1. **Sprint 1 - Set up project and user interface:**
    - **Task 1: Create a project plan and backlog**
    - **Task 2: Implement user interface**
    - **Task 3: Develop user registration and login functionality**

Code:

multiple_disease_prediction01.py

F: > python code > temporary - Copy > webapp > multiple_disease_prediction01.py

```python
import streamlit as st
import pandas as pd
import pickle
from streamlit_option_menu import option_menu

import json
from PIL import Image
import requests  # pip install requests
import streamlit as st  # pip install streamlit
from streamlit_lottie import st_lottie  # pip install streamlit-lottie

# GitHub: https://github.com/andfanilo/streamlit-lottie
# Lottie Files: https://lottiefiles.com/

# loading the saved models

diabetes_model = pickle.load(open('F:/python code/temporary/model/diabetes_trained_model.sav', 'rb'))

heart_disease_model = pickle.load(open('F:/python code/temporary/model/heart_disease_model.sav','rb'))

parkinsons_model = pickle.load(open('F:/python code/temporary/model/parkinsons_model.sav', 'rb'))

lung_cancer_model = pickle.load(open('F:/python code/temporary/model/lungCancer_trained_model.sav', 'rb'))

```



2. Sprint 2 - Implement symptom checker and disease prediction, algorithm

- **Task 1: Develop a disease prediction algorithm**
- **Task 2: Implement model training and hyperparameter tuning.**
- **Task 3: Test symptom checker and prediction algorithm.**

Code:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score

# loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('/content/diabetes.csv')

# getting the statistical measures of the data
diabetes_dataset.describe()

# separating the data and labels
X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

Train and test Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)
#Training model

classifier = svm.SVC(kernel='linear')
#training the support vector Machine Classifier
classifier.fit(X_train, Y_train)
# accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
# accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
```

3. **Sprint 3 - Disease prediction information pages**

   - **Task 1: Disease prediction information pages**
   - **Task 2: Deploy the disease prediction system in a production environment and Make a Predictive System**

Code:

C: > Users > User > New Text Document.py

```
    choice = st.sidebar.selectbox("Menu",menu)

    elif choice == "Login":
        st.subheader("Login Section")
        username = st.sidebar.text_input("User Name")
        password = st.sidebar.text_input("Password",type='password')
        if st.sidebar.checkbox("Login"):
            # if password == '12345':
            create_usertable()
            hashed_pswd = make_hashes(password)

            result = login_user(username,check_hashes(password,hashed_pswd))
            if result:

                st.success("Logged In as {}".format(username))

                selected = option_menu('',")
                # Diabetes Prediction Page
                if (selected == 'Diabetes Prediction'):
                    # page title
                    img = Image.open("F:/python code/temporary/image/1.webp")
                    st.image(img)
                    st.title('Diabetes Prediction using ML')

                # Heart Disease Prediction Page
                if (selected == 'Heart Disease Prediction'):
                    img = Image.open("F:/python code/temporary/image/2.jpg")
                    st.image(img)
                    # page title
                    st.title('Heart Disease Prediction using ML')

                # parkinsons Prediction Page
```

4. **Sprint 4 - Improve user experience and performance**

   - **Task 1: Optimize page load times**
   - **Task 2: Improve UI and user experience**
   - **Task 3: Test performance and user experience improvements**

Code:

```
input_data = (5,166,72,19,175,25.8,0.587,51)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
print(std_data)

prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

Project developing resource

Model:

The analysis of different disease datasets involved the use of machine-learning classification algorithms. They were as follows:

- Support Vector Machine (Linear SVM)
- Logistic Regression

Support Vector Machine:

A Support Vector Machine (SVM) is a type of tool that helps classify data into different groups. It works by drawing a boundary between the different groups, trying to make the distance between them as big as possible. This boundary is called a hyperplane and the points closest to it are called support vectors. SVMs are useful for handling lots of data and dealing with data that isn't perfect. They can be used in many areas, such as image classification, text classification, and biology.

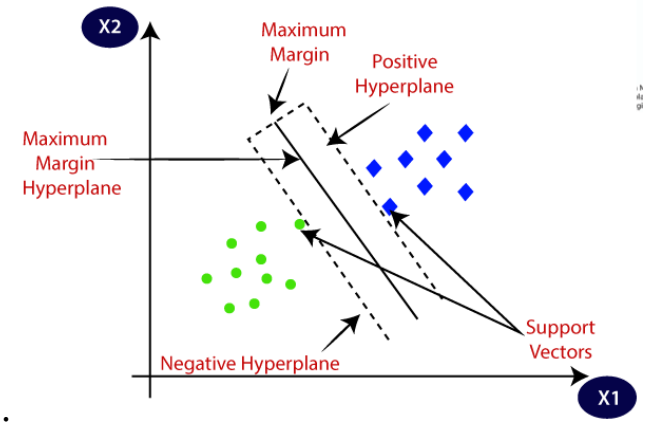


Figure: Linear SVM

Logistic Regression:

Logistic Regression is a machine learning algorithm used for classification problems. It works by finding the best line or curve that can separate two groups of data. It then calculates the probability of a new data point belonging to each group based on its distance from the line or curve. This algorithm is commonly used in applications such as predicting if a customer will buy a product or not based on their past behavior, or determining if a patient is likely to have a certain disease based on their symptoms.

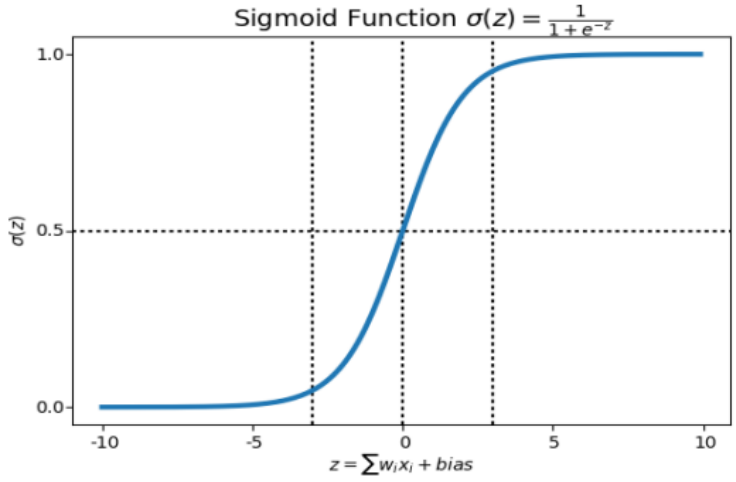


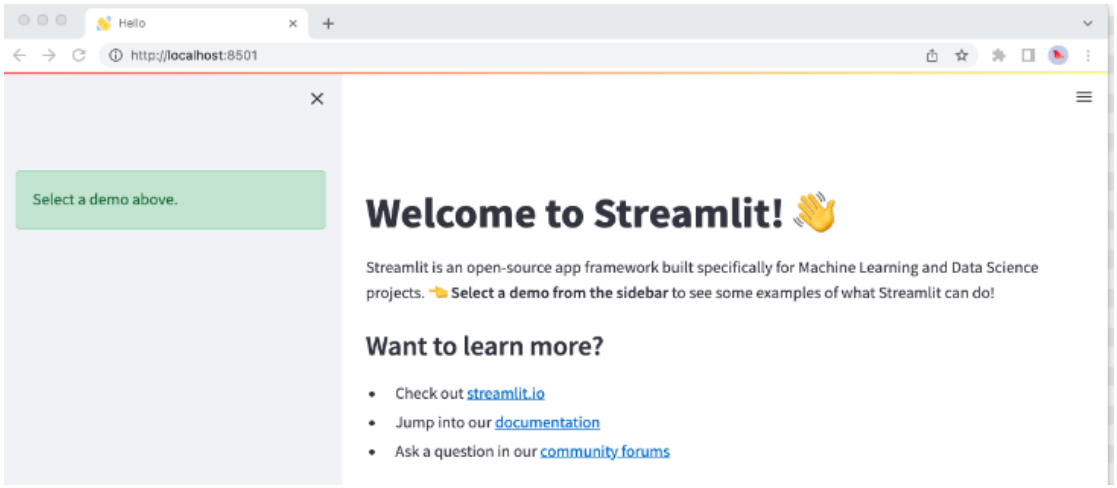
Figure: sigmoid function graph

Design pattern:

Here we use an MVC design pattern. As I mentioned earlier, the Model-View-Controller (MVC) pattern could be a useful design pattern for structuring the application.

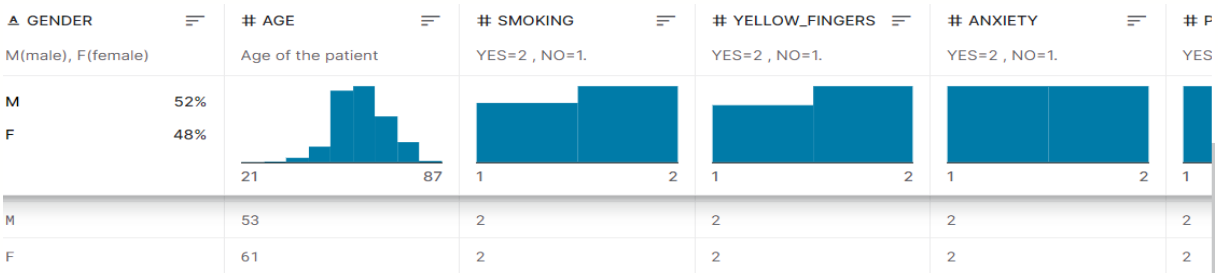
Framework:

Streamlit is a Python-based web framework that allows developers to quickly build and deploy data-driven web applications. You can use Streamlit to create a disease prediction web app by defining the app interface, loading data, running the disease prediction algorithm, and displaying the results. Once your app is built, you can deploy it to a web server or cloud platform using Streamlit.



Dataset:

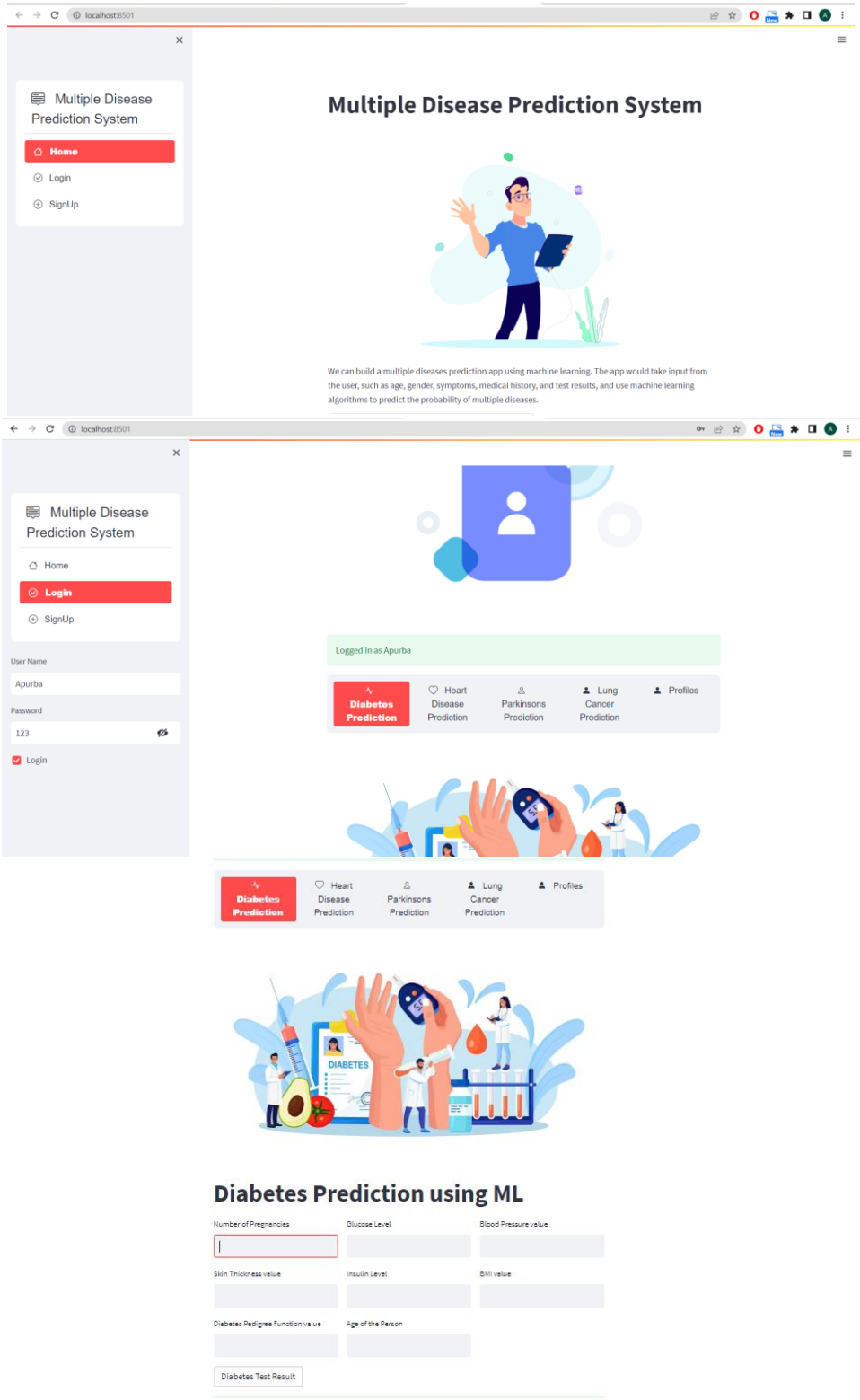
The project is to use machine learning algorithms to classify and predict diseases using various datasets available on Kaggle.



- **Diabetics prediction dataset:** https://www.kaggle.com/datasets/kandij/diabetesdataset
- **Heart Disease prediction dataset:** https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
- **Parkinson's Disease dataset:** https://www.kaggle.com/datasets/gargmanas/parkinsonsdataset
- **Lung Cancer dataset:** https://www.kaggle.com/datasets/mysarahmadbhat/lung-cancer

Final Result of Project:

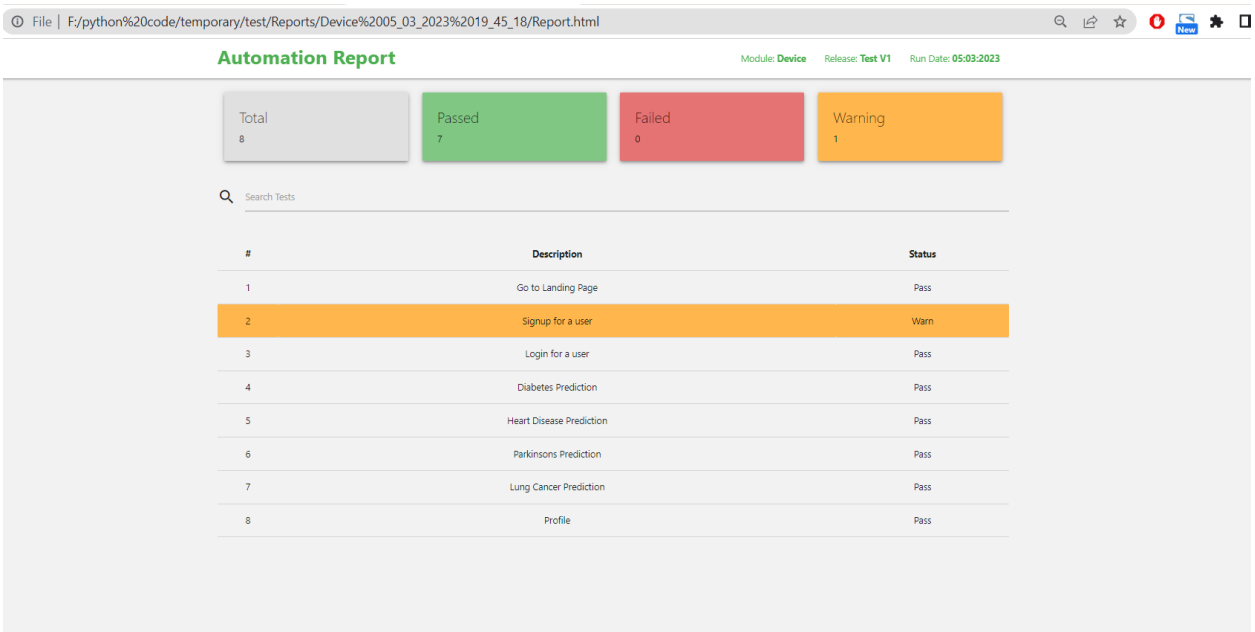
Some sample images of the project



Testing script result:

```
import random
import time
from telnetlib import EC
from pyhtmlreport import Report
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.webdriver import WebDriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait

report = Report()
service = Service("F:\chromedriver\chromedriver.exe")
driver = webdriver.Chrome(service=service)
#driver: WebDriver = webdriver.Chrome(executable_path="F:\chromedriver\chromedriver.exe")
report.setup(
report_folder=r'Reports',
module_name='Device',
release_name='Test V1',
selenium_driver=driver)
driver.get('http://192.168.31.220:8501')
time.sleep(5)

# Test Case 01
try:
    report.write_step(
    'Go to Landing Page',
    status=report.status.Start,
    test_number=1)
    #assert (driver.title == 'multiple_disease_prediction · Streamlit')
    report.write_step(
    'Landing Page loaded Successfully.',
    status=report.status.Pass,
    screenshot=True)
except AssertionError:
    report.write_step(
    'Landing Page loaded Successfully.',
    status=report.status.Fail,
    screenshot=True)
except Exception as e:
    report.write_step(
    'Something went wrong!</br>{e}',
    status=report.status.Warn,
    screenshot=True)
```

```
# Test Case 02
try:
    report.write_step(
        'Signup for a user',
        status=report.status.Start,
        test_number=2)
    driver.find_element(By.XPATH, '/html/body/div/div/div/div/div/ul/li[3]/a').click()
    driver.find_element(By.XPATH,'/html/body/div[1]/div[1]/div[1]/div/div/div/section[2]/div[1]/div[1]/div/div[3]/div/div[1]/div/input').send_keys('apu')
    driver.find_element(By.XPATH,'/html/body/div[1]/div[1]/div[1]/div/div/div/section[2]/div[1]/div[1]/div/div[4]/div/div[1]/div/input').send_keys('123')
    # Click on the button
    button = WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.XPATH, '/html/body/div/div[1]/div[1]/div/div/div/section[2]/div[1]/div[1]/div/div[5]/div/button')))
    button.click()
    report.write_step(
        'Successfully Signup ',
        status=report.status.Pass,
        screenshot=True )
```

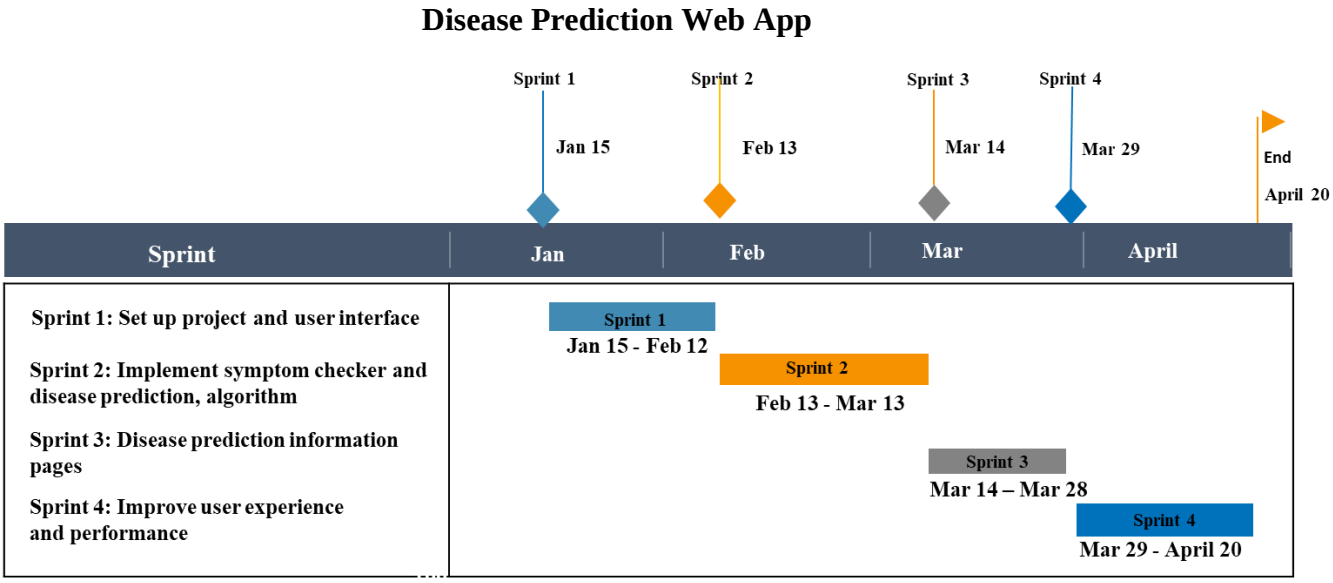


Project management

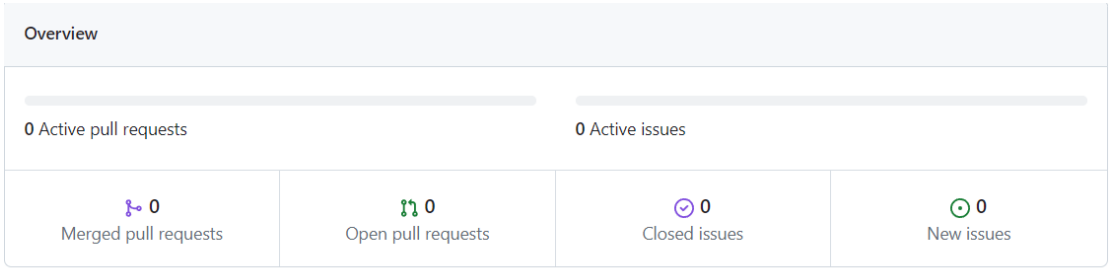
Project Timeline (According to the process model):

Sprint	**Sprint 1:** Set up project and user interface	**Sprint 2:** Implement symptom checker and disease prediction, algorithm	**Sprint 3:** Disease prediction information pages	**Sprint 4:** Improve user experience and performance
Start Date To End Date	15-01-2023 To 12-02-2023	13-02-2023 To 13-03-2023	14-03-2023 To 28-03-2023	29-03-2023 To 20-04-2023
Duration	28 4 weeks	28 4 weeks	14 2 weeks	21 3 weeks

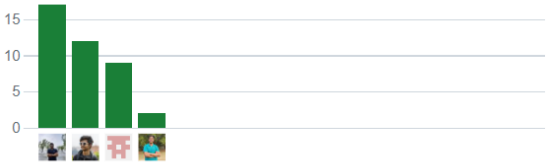
Project Grant chart:

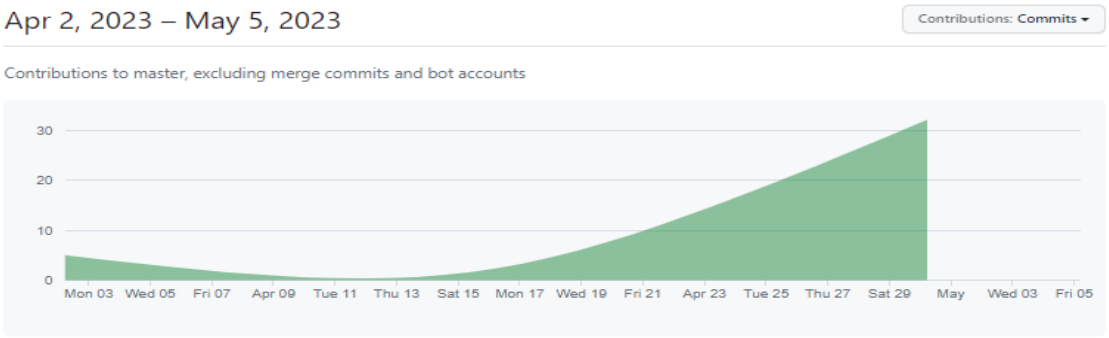


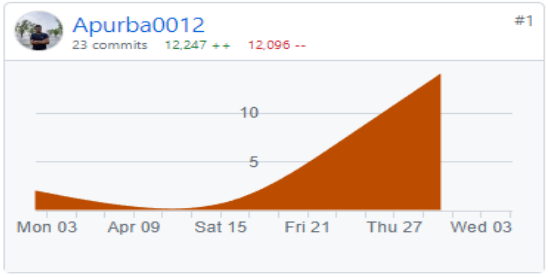
Version Control System commits snapshot:

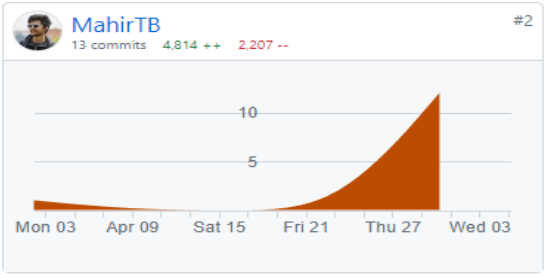


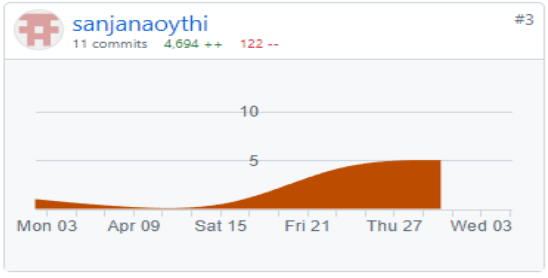
Excluding merges, **4 authors** have pushed **40 commits** to master and **40 commits** to all branches. On master, **46 files** have changed and there have been **8,246** additions and **289** deletions.











Finance management (According to the project Module):

Project cost:

Task	Star Time	Finish Time	Duration
Task 01: Set up project and user interface	15-01-2023	12-02-2023	28
Task 02: Implement symptom checker and disease prediction, algorithm	13-02-2023	13-03-2023	28
Task 03: Disease prediction information pages	14-03-2023	28-03-2023	14
Task 04: Improve user experience and performance	29-03-2023	20-04-2023	21

Total number of working days for the project = 28 + 28 + 14 + 21 = 91 days

Total number of working hours per day per software engineer = 5 hours

Total number of working hours per day for all 4 software engineers = 5 hours * 4 = 20 hours

Total number of working hours for the project = Total number of working days for the project * Total number of working hours per day

for all 4 software engineers = 91 days * 20 hours = 1820 hours

In Bangladesh average software engineer's monthly salary = 45000tk.S0, Hourly salary per software engineer = 300tk/hour

Total cost for all 4 software engineers per hour = 300tk/hour * 4 software engineers = 1200tk/hour

Total project cost = Total number of working hours for the project * Total cost for all 4 software engineers per hour = 1820 hours * 1200tk/hour = 2,184,000tk

Therefore, the total cost of the project with a 5-hour workday and hourly salary of 300tk for 4 software engineers working on the project is 2,184,000tk.

Conclusion/ Future work:

This project is an excellent demonstration of how technology can be used to improve healthcare by providing a quick and accurate prediction of diseases. Through this project, you have gained valuable experience in software development, web development, and machine learning. You have also learned about different technologies and frameworks such as Flask, scikit-learn, and HTML/CSS/JS, streamlit which can be used to develop web applications.

Here are some key takeaways and future work recommendations based on your project:

1. **Project management:** Effective project management is essential for delivering a successful project. Make sure to establish clear goals, timelines, and milestones. Also, ensure effective communication among team members.
2. **Distributed and collaborative software development:** It's essential to understand how to work effectively in a distributed and collaborative software development environment. Ensure that everyone on the team has access to the necessary tools and resources, and establish communication.
3. **Risk analysis:** It's important to identify potential risks early in the project and develop a plan to address them. This will help you avoid costly delays and setbacks down the line.

As for future work, here are some areas you might consider exploring:

1. **User feedback:** Get feedback from users to identify areas for improvement and potential new features to add.
2. **Integration with other systems:** Consider integrating your Disease Prediction Web App with other health-related systems, such as electronic medical records.
3. **Machine learning:** Investigate the use of machine learning algorithms to improve the accuracy of disease prediction.

Overall, the Disease Prediction Web App project provided a valuable learning experience in project management, distributed and collaborative software development, and risk analysis. Keep building on experience to develop even more complex software-intensive systems in the future.

Reference:

[1] Meisam Shabanpoor, Mehregan Mahdavi, "Application of recommender systems on disease recognition and treatment", ResearchGate, July 2011

[2] "Multiple Disease Prediction Webapp", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and ISSN Approved), ISSN:2349-5162, Vol.9, Issue 10, page no. ppe225-e234, October-2022, Available at : http://www.jetir.org/papers/JETIR2210432.pdf

[3] Md. Tahmid Rahman Laskar, Md. Tahmid Hossain, Abu Raihan Mostofa Kamal, Nafiul Rashid, "Automated Disease Prediction System (ADPS): A User Input-based Reliable Architecture for Disease Prediction", ResearchGate, International Journal of Computer Applications (IJCA), Vol. 133- No.15, January 2016

[4] Kedar Pingale1, Sushant Surwase2, Vaibhav Kulkarni3, Saurabh Sarage4, Prof. Abhijeet Karve5, "Disease Prediction Using Machine Learning", International Research Journal of Engineering and Technology (IRJET), Vol. 06, Issue 12, December 2019

[5] Ayon Dey, "Machine Learning Algorithms: A Review", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 7 (3), 1174-1179, 2016

[6] Archit Verma, "STUDY AND EVALUATION OF CLASSIFICATION ALGORITHMS IN DATA MINING", International Research Journal of Engineering and Technology (IRJET), Volume: 05 Issue: 08, August 2018

CEP Mapping

How Ks are addressed through the project and mapping among Ks, COs, and POs

Ks	Attributes	How K's are addressed	COs	POs
K3	Engineering fundamentals	Data collection, Python, Machine Learning	CO1	PO(a)
K4	Specialist knowledge	Steamlit framework, Machine learning model (Support Vector Machine, Logistic Regression model)	CO1	PO(a)
K5	Engineering Design	UML, MVC, Agile methodology, UI/UX Design.	CO3	PO(c)
K6	Engineering practice	Pycharm, Anaconda, Pandas, numPy, Streamit, Python libraries for Machine Learning (Logistic Regression)	CO4	PO(e)

How Ps are addressed through the project and mapping among Ps, COs, and POs

Ps	Attributes	How P's are addressed	COs	POs
P1	Depth of knowledge required	Requires knowledge about Streamlit for developing the web app. (K4) Project requires UML and MVC framework for designing, Agile methodologies to design and analyze the whole project. (K5)	CO1 CO3	PO(a) PO(c)

		We need to have proper knowledge about Python libraries (Pandas, numPy) for Machine Learning (K6).		
P3	Depth of analysis required	The project requires studying more SVM models. Also more knowledge about Logistic Regression is required.to make the prediction system more accurate.	CO2 CO8	PO(b) PO(j)
P7	Interdependence	• Data collection and processing • Machine learning model development • Model deployment and predicting system	CO3 CO7 CO8	PO(c) PO(i) PO(j)

How As are addressed through the project

As	Attributes	How A's are addressed	COs	POs
A1	Range of resources	The project needs various medical data and other information and technologies.	CO4 CO6	PO(e) PO(h)
A4	Consequences for society and environment	Our work helps the people to obtain an immediate result of disease type hence helps them to take proper medication in time.	CO5	PO(f)