

# CSCE 5214.004 SDAI

## Group 9: DIABETES PREDICTION TOOL

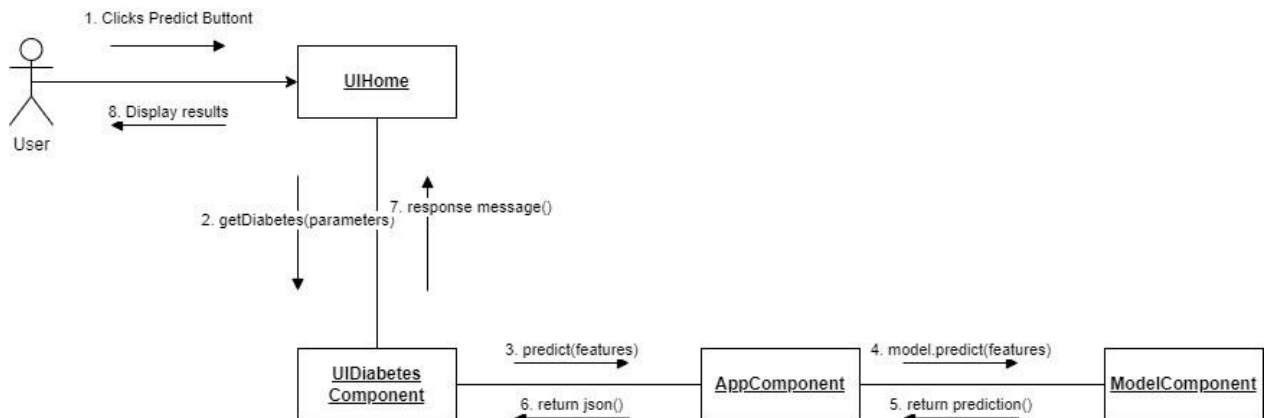
### Phase 4: Develop a User Interface

Aditya Vadrevu – 11601517  
Suhassiddharajgari Tellatakula – 11626111  
Srikanth Reddy Gunna - 11594012  
Sai Praneeth Reddy Avula - 11582402  
Indu Shashi Guda - 11618418

#### 1. Task 1: Communication Diagram:

##### Use Case 1:

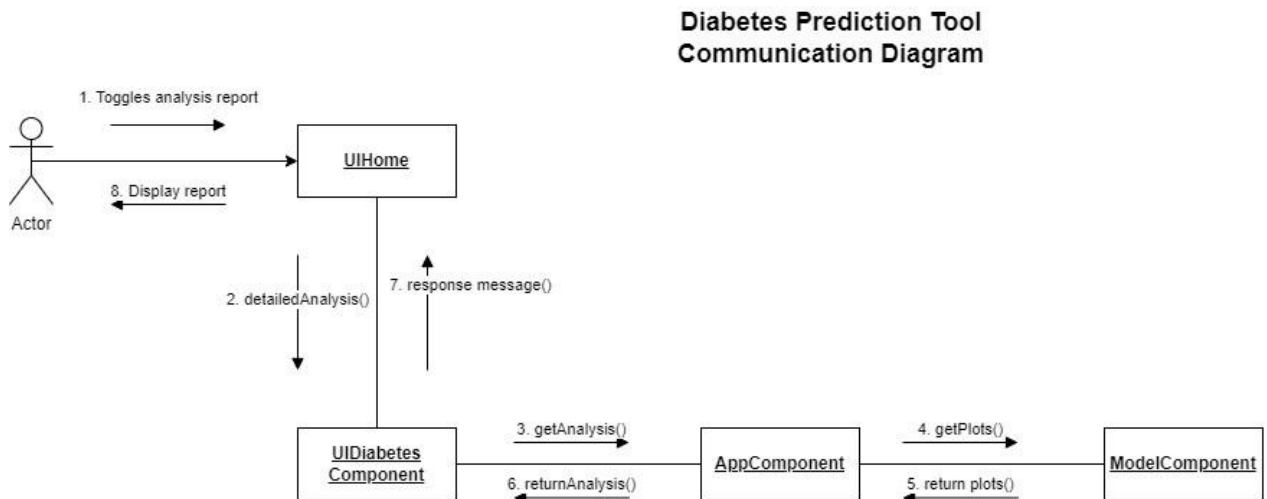
Diabetes Prediction Tool  
Communication Diagram



Above communication diagrams depicts the Predict Use Case. When the user provides his values and clicks on the predict button, the html component in UI Home container will pass the entered values as arguments through getDiabetes() method. This method which is defined in UI Diabetes component whenever called , will initiate http.get along with the received parameters namely glucose level, Insulin, BMI and Age. In the backend code of the tool in App.py component, function called predict will initiate whenever get method is called. It collects these 4 parameters and convert them using minmaxscalar and pass it to the backend finalized model which is already dumped and saved using joblib

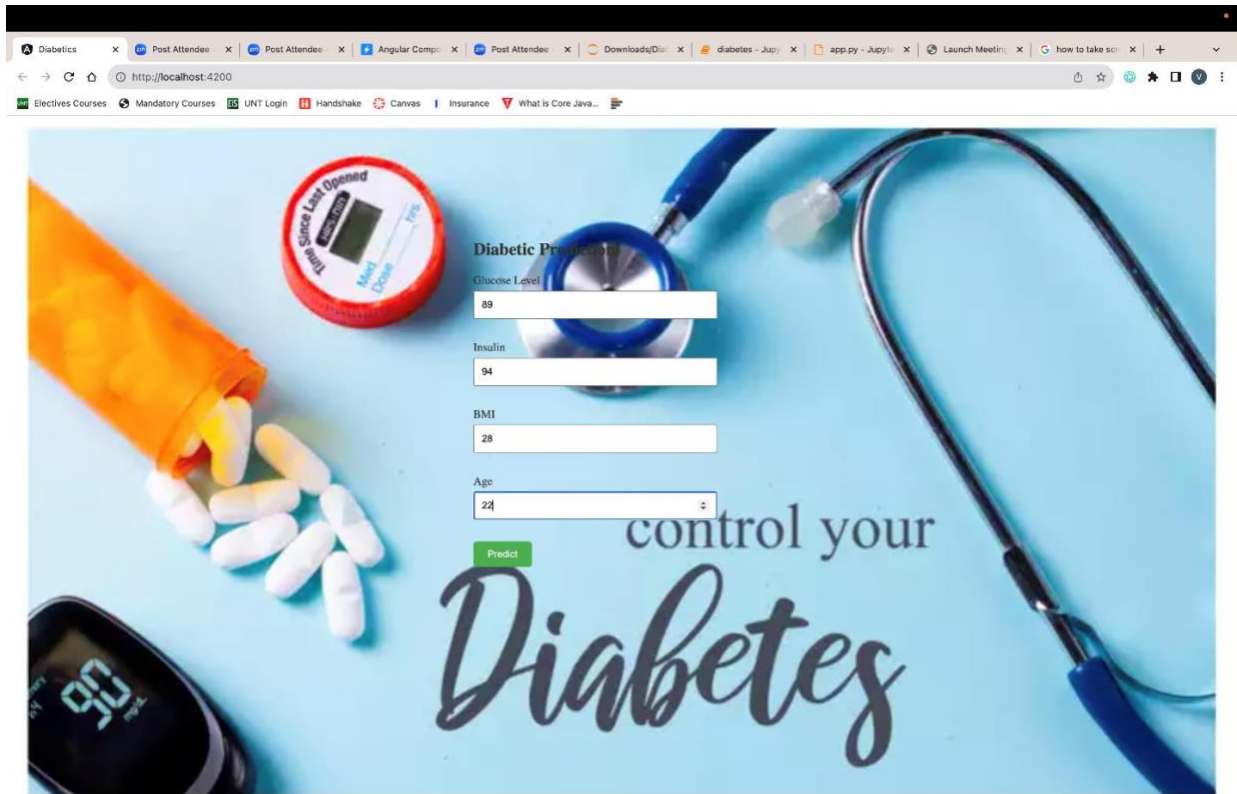
library. So , this model will take these values and predicts the output for these specific values. If the output is 0, “You don’t have Diabetes” message is assigned. Otherwise, “You have Diabetes. Please consult a doctor” message is assigned . This assigned message is returned in the form of json to the UI diabetes component which will then display the response message to user through UI Home component html web page.

## **Use Case 2:**



Above communication diagrams depicts the Detailed Analysis Report Use Case. Once the user can see his/her results on the results page, he can toggle the detailed analysis report which will initiate the detailedAnalysis() method in the html component - UI Home container which will in turn call getAnalysis() in UIDiabetes Component. This method which is defined in UI Diabetes component whenever called , will initiate http.get. In the backend code of the tool in App.py component, function called getplots() will initiate whenever get method is called. Then the model will process and return the plots to app and UI Diabetes component which will then display the response message to user through UI Home component html web page.

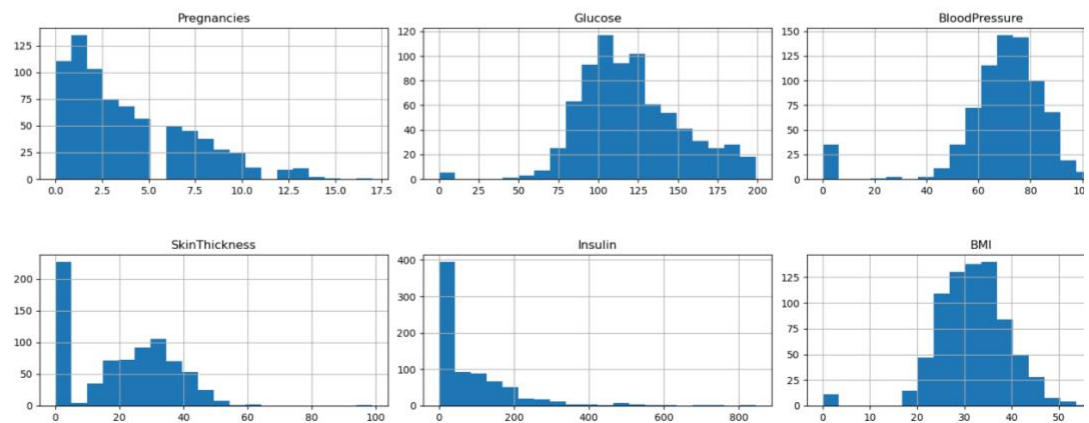
## 2. Task 2: UI Implementation

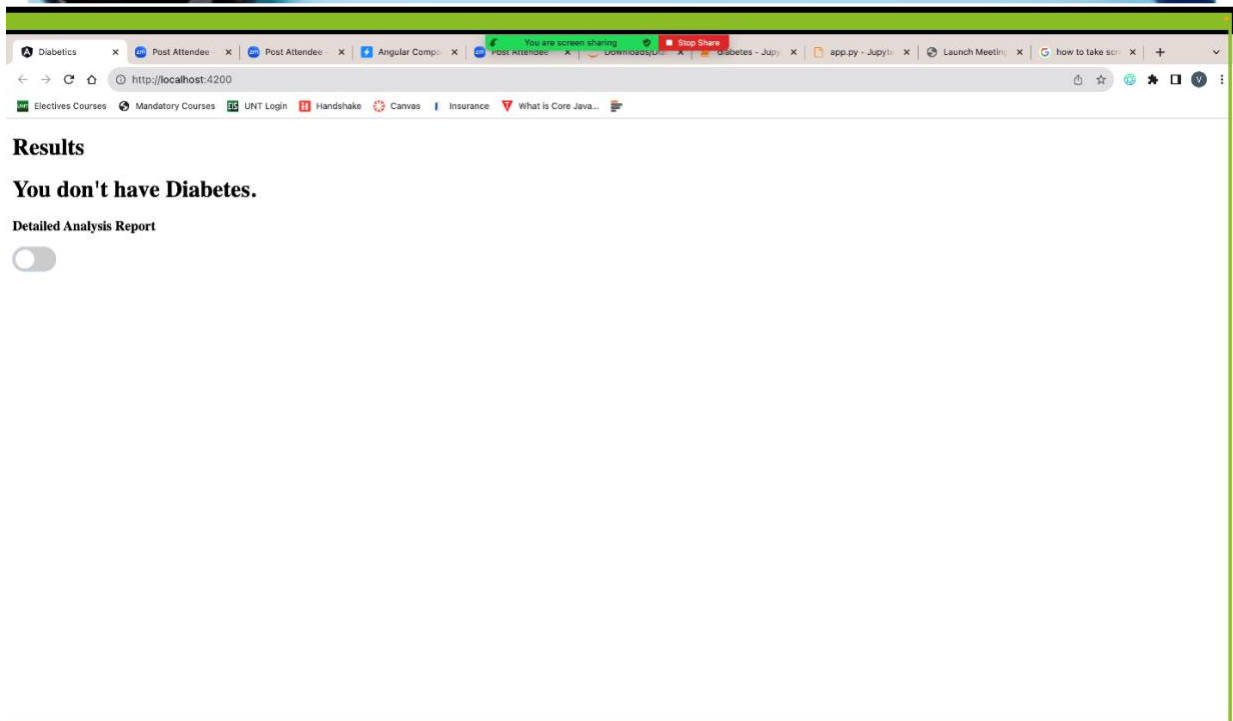
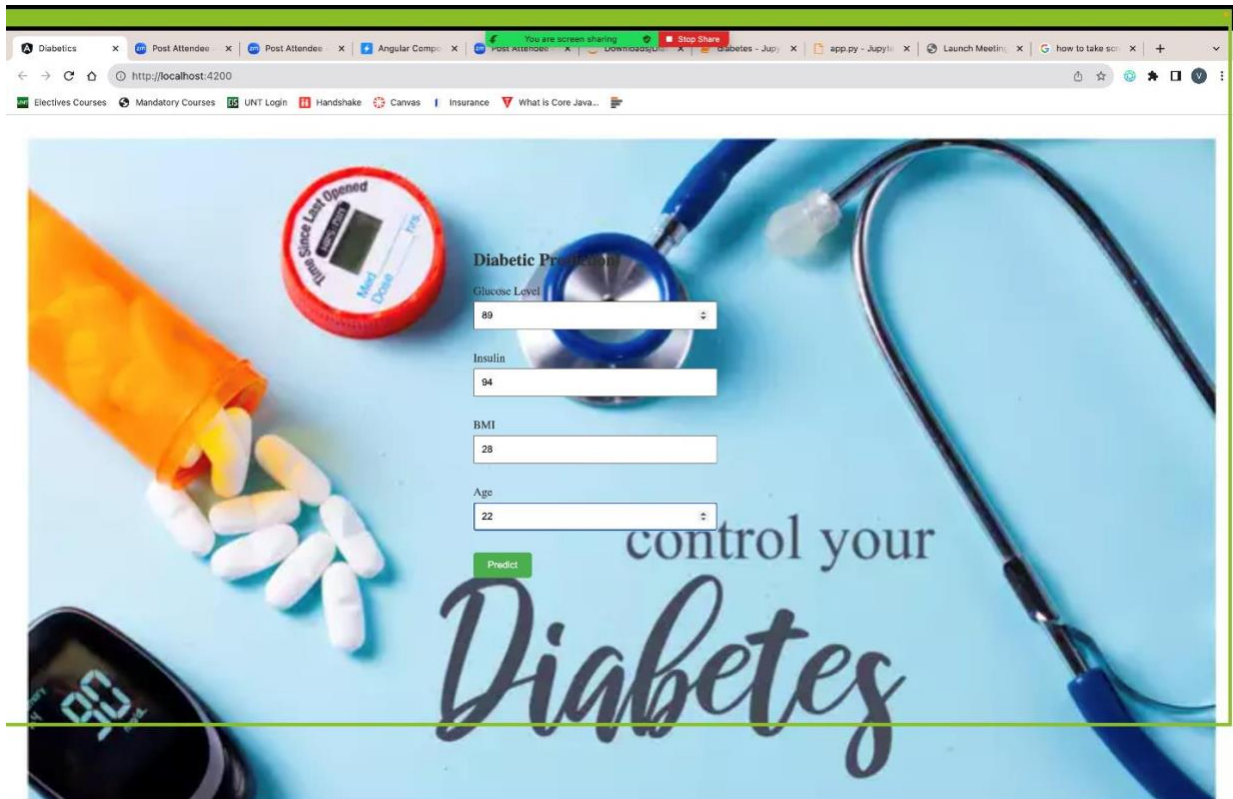


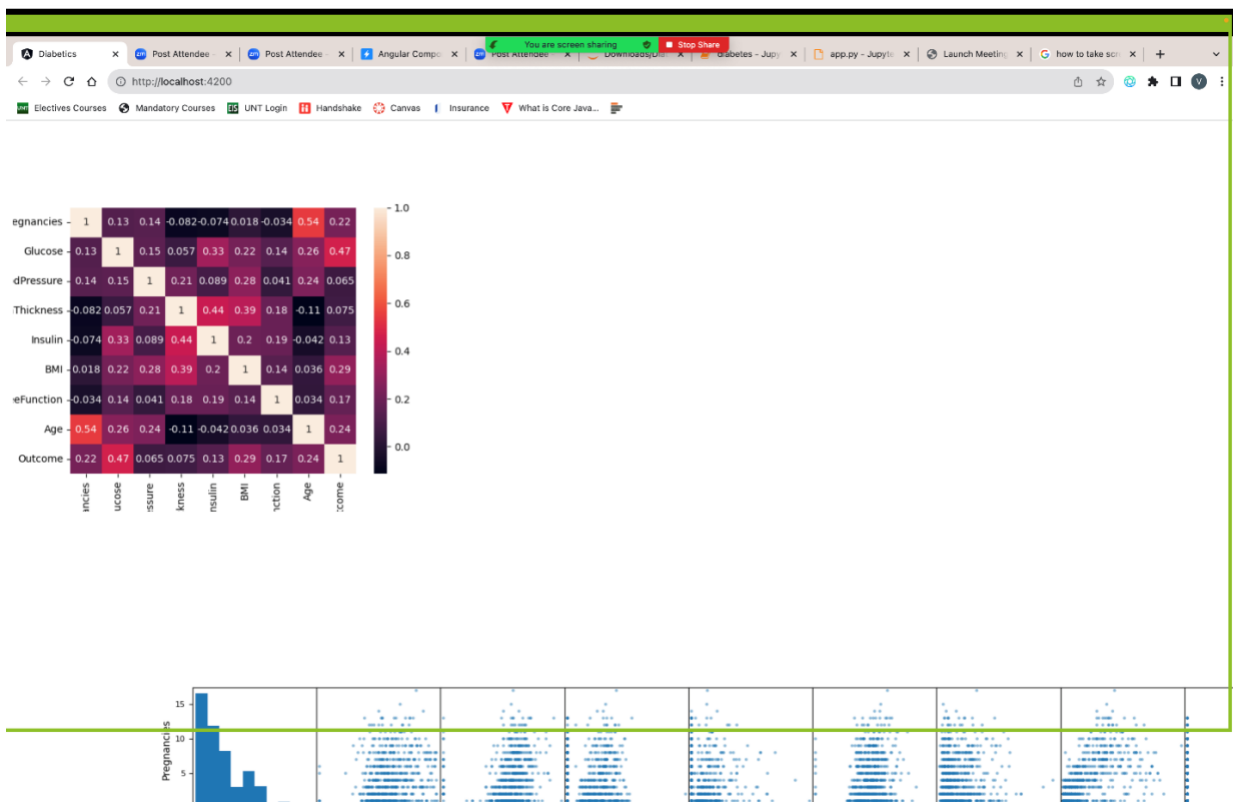
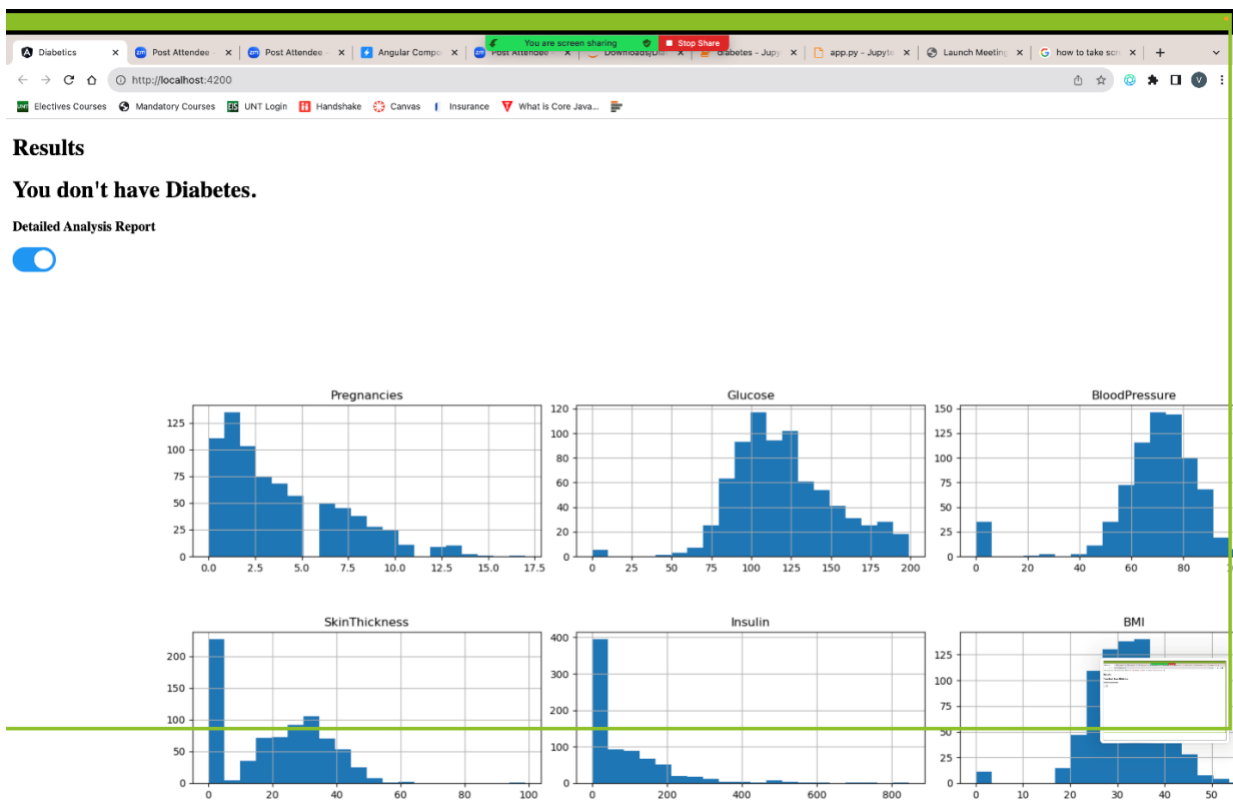
### Results

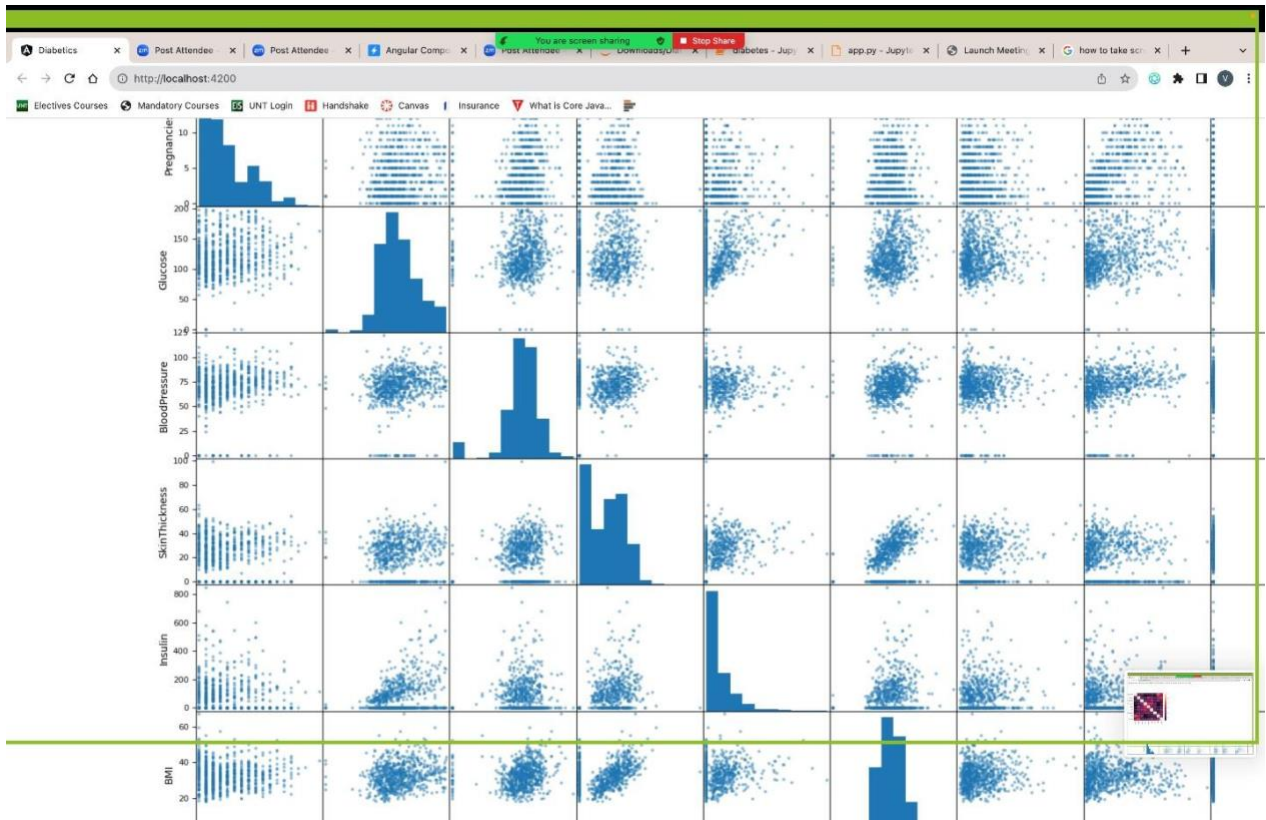
**You don't have Diabetes.**

Detailed Analysis Report



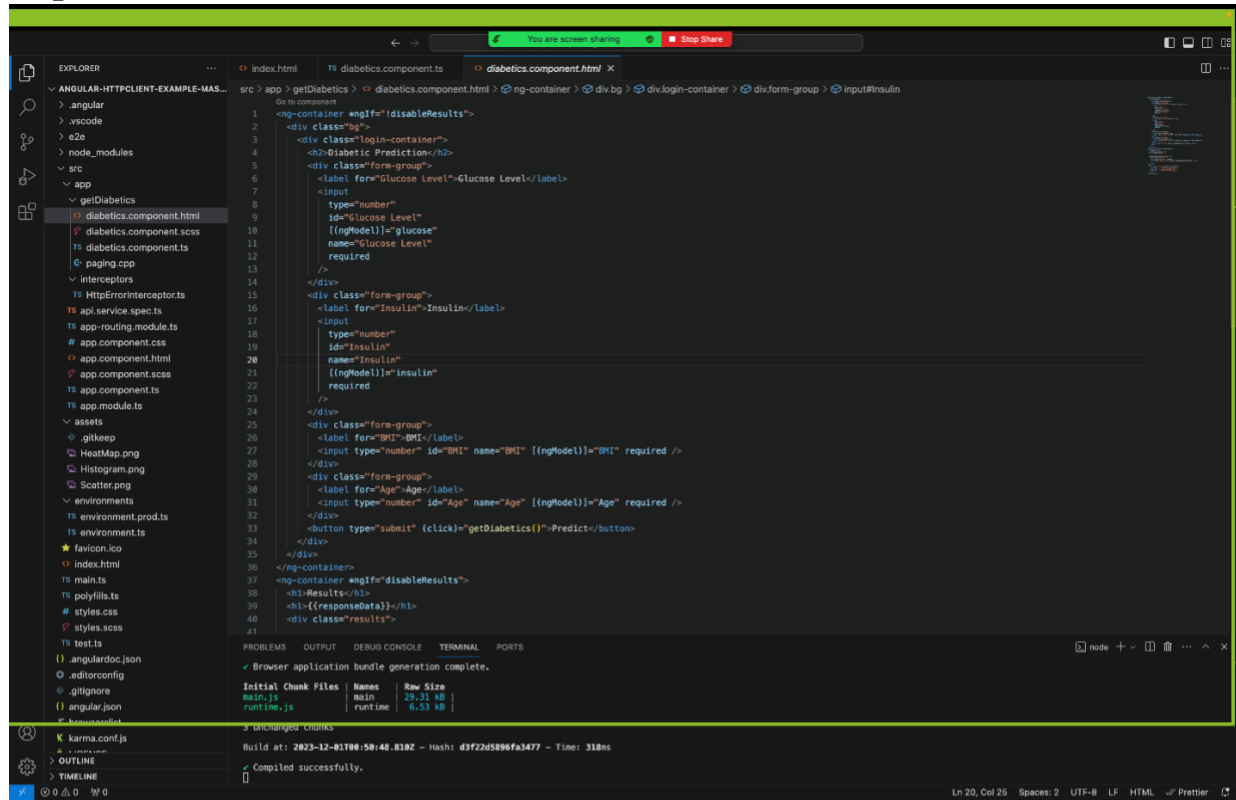






Above figures show the User Interface implemented additional to the existing Diabetes Prediction tool developed. In the earlier version, it has a very basic and minimal UI which used to take values from user and return yes/no result as pop-up. So, we have enhanced this system by developing a new UI using Angular, HTML, CSS. When the user opens the tool, he will land in the home page where he/she can enter the values like Glucose level, Insulin, BMI and Age with a relevant background image as shown above. Also, there is a predict button. Then we have a results page which shows the response message. Also, we added a detailed analysis report which shows the data correlation amongst features and the output in terms of scatter plots, heatmaps, histograms. This can be viewed using a toggle button as shown above.

## Explanation of code:



```
1 <ng-container *ngIf="disableResults">
2   <div class="bg">
3     <div class="login-container">
4       <h2>Diabetic Prediction</h2>
5       <div class="form-group">
6         <label for="Glucose Level">Glucose Level</label>
7         <input
8           type="number"
9           id="Glucose Level"
10          [(ngModel)]="glucose"
11          name="Glucose Level"
12          required
13        />
14      </div>
15      <div class="form-group">
16        <label for="Insulin">Insulin</label>
17        <input
18          type="number"
19          id="Insulin"
20          name="Insulin"
21          [(ngModel)]="insulin"
22          required
23        />
24      </div>
25      <div class="form-group">
26        <label for="BMI">BMI</label>
27        <input type="number" id="BMI" name="BMI" [(ngModel)]="BMI" required />
28      </div>
29      <div class="form-group">
30        <label for="Age">Age</label>
31        <input type="number" id="Age" name="Age" [(ngModel)]="Age" required />
32      </div>
33      <button type="submit" (click)="getDiabetics()">Predict</button>
34    </div>
35  </ng-container>
36  <ng-container *ngIf="disableResults">
37    <h3>Results</h3>
38    <h3>{{responseData}}</h3>
39    <div class="results">
40
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

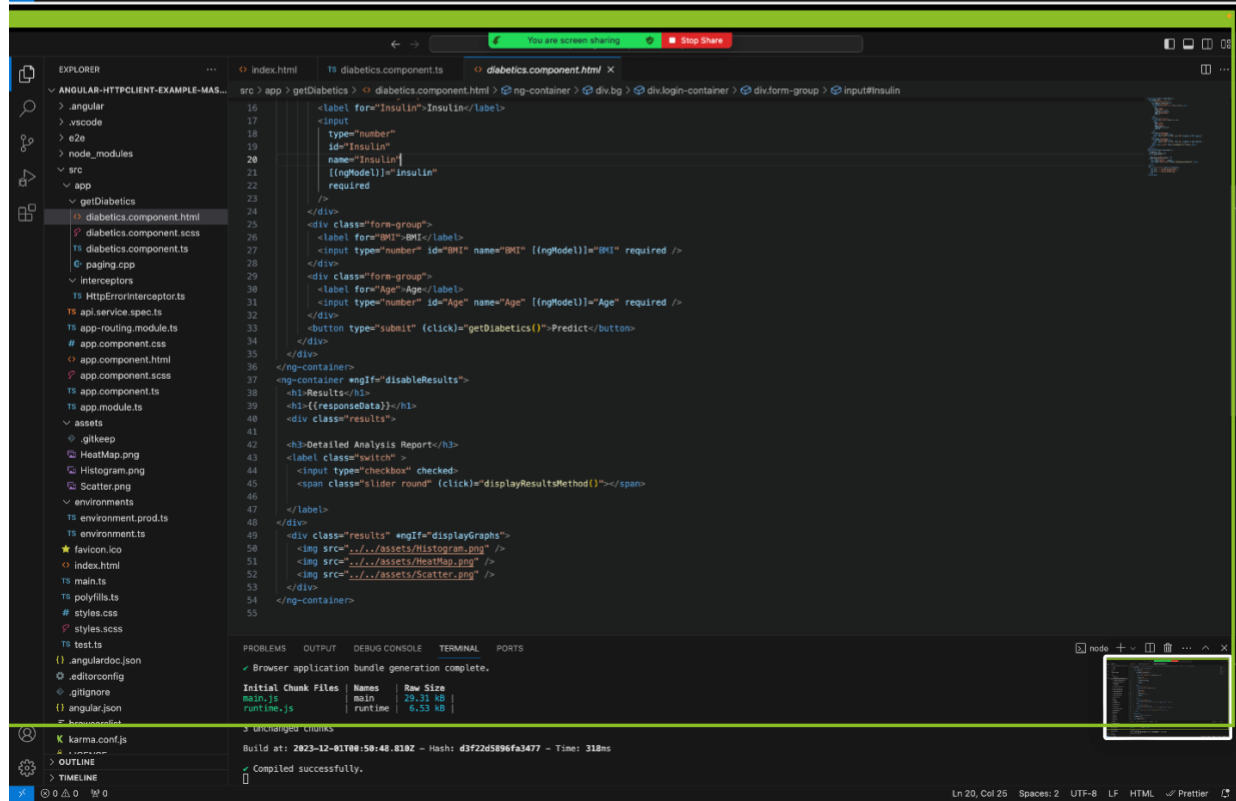
✓ Browser application bundle generation complete.

Initial Chunk Files	Names	Raw Size
main.js	main	28.3 KB
runtime.js	runtime	6.53 KB

Build at: 2023-12-01T06:50:48.010Z - Hash: d3f22d5896fa3477 - Time: 318ms

✓ Compiled successfully.

Ln 20, Col 26 Spaces: 2 UTF-8 LF HTML Prettier



```
41
42    <h3>Detailed Analysis Report</h3>
43    <label class="switch">
44      <input type="checkbox" checked="" />
45      <span class="slider round" (click)="displayResultsMethod()"></span>
46    </label>
47  </div>
48  <div class="results" *ngIf="displayGraphs">
49    
50    
51    
52  </div>
53  </ng-container>
54
55
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ Browser application bundle generation complete.

Initial Chunk Files	Names	Raw Size
main.js	main	28.3 KB
runtime.js	runtime	6.53 KB

Build at: 2023-12-01T06:50:48.010Z - Hash: d3f22d5896fa3477 - Time: 318ms

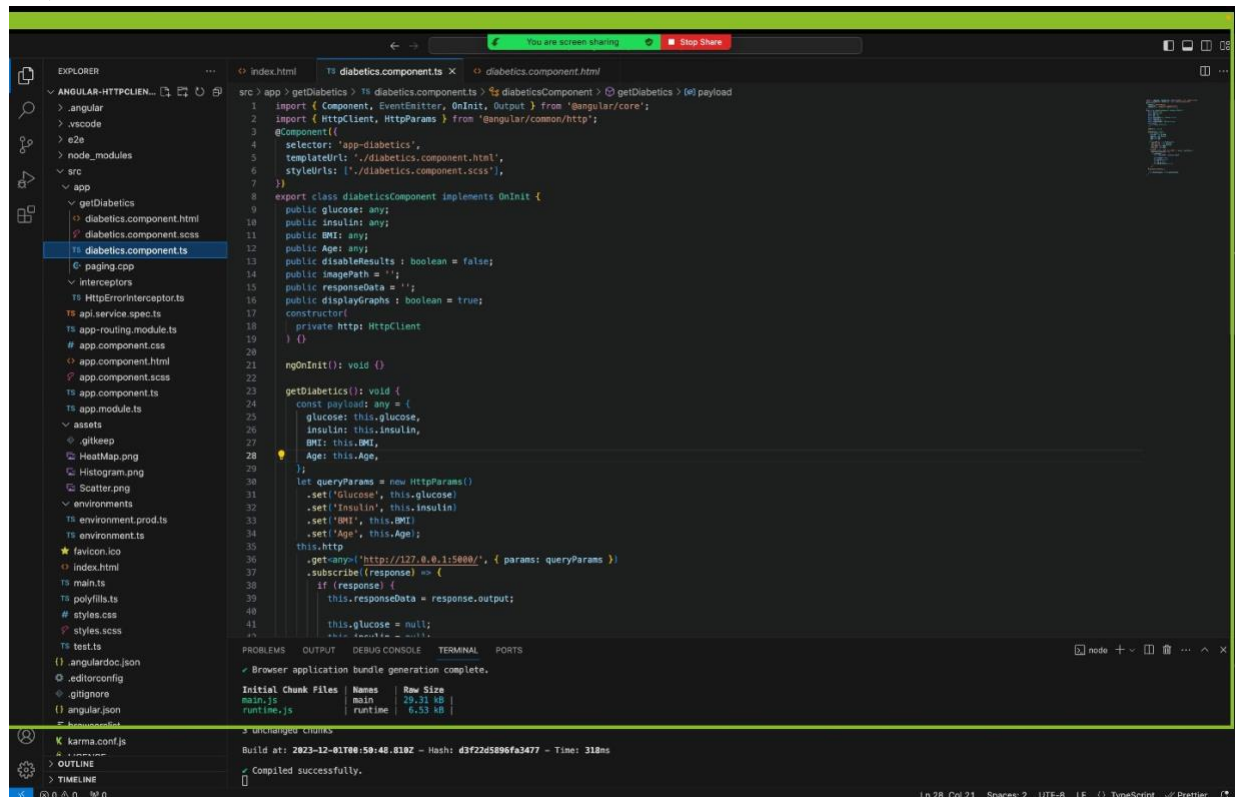
✓ Compiled successfully.

Ln 20, Col 26 Spaces: 2 UTF-8 LF HTML Prettier



Above 2 screenshots the diabetes component html code which has both home and results webpage. Home page contains placeholders for Glucose level, Insulin, BMI and Age. Whenever the predict button is clicked, `getDiabetes ()` method is called as shown in line number 33.

We can see the `getDiabetics()` function defined in the screenshot below where the entered values are passed to the backend through `http.get` method as seen in lines 35,36.



```
src > app > getDiabetics > 18 diabetics.component.ts > 18 diabeticsComponent > 18 getDiabetics > 18 payload
1 import { Component, EventEmitter, OnInit, Output } from '@angular/core';
2 import { HttpClient, HttpParams } from '@angular/common/http';
3 @Component({
4   selector: 'app-diabetics',
5   templateUrl: './diabetics.component.html',
6   styleUrls: ['./diabetics.component.scss'],
7 })
8 export class diabeticsComponent implements OnInit {
9   public glucose: any;
10  public insulin: any;
11  public BMI: any;
12  public Age: any;
13  public disableResults: boolean = false;
14  public imagePath = '';
15  public responseData = '';
16  public displayGraphs: boolean = true;
17  constructor(
18    private http: HttpClient
19  ) {}
20  ngOnInit(): void {}
21
22  getDiabetics(): void {
23    const payload: any = {
24      glucose: this.glucose,
25      insulin: this.insulin,
26      BMI: this.BMI,
27      Age: this.Age,
28    };
29    let queryParams = new HttpParams();
30    .set('Glucose', this.glucose);
31    .set('Insulin', this.insulin);
32    .set('BMI', this.BMI);
33    .set('Age', this.Age);
34    this.http
35      .get('http://127.0.0.1:5000/', { params: queryParams })
36      .subscribe((response) => {
37        if (response) {
38          this.responseData = response.output;
39          this.glucose = null;
40          this.insulin = null;
41          this.BMI = null;
42          this.Age = null;
43        }
44      });
45  }
```

Initial Chunk Files	Name	Raw Size
main.js	main	20.31 kB
runtime.js	runtime	6.53 kB

Build at: 2023-12-01T00:59:48.018Z - Hash: d3f22d5896fa3477 - Time: 318ms  
Compiled successfully.

This will initiate the predict block in backend `app.py` in below screenshot because it starts whenever GET is called. Inside this predict, These 4 parameters are converted into numpy array features and passed to the `model.predict`. We can notice that this model is already dumped and saved then called here. If the prediction turns out to be 1, "You don't have diabetes" message is returned. If it is 0, "You have diabetes, please consult a doctor" message is returned. This message is returned as json.



Diabetics x Post Attendee x Post Attendee x Angular Comp: x You are screen sharing x Stop Share x app.py - Jupy: x Launch Meetin: x how to take sci: x +

http://localhost:8888/edit/Downloads/Diabetes-Prediction/flask/app.py

Electives Courses Mandatory Courses UNT Login Handshake Canvas Insurance What is Core Java...

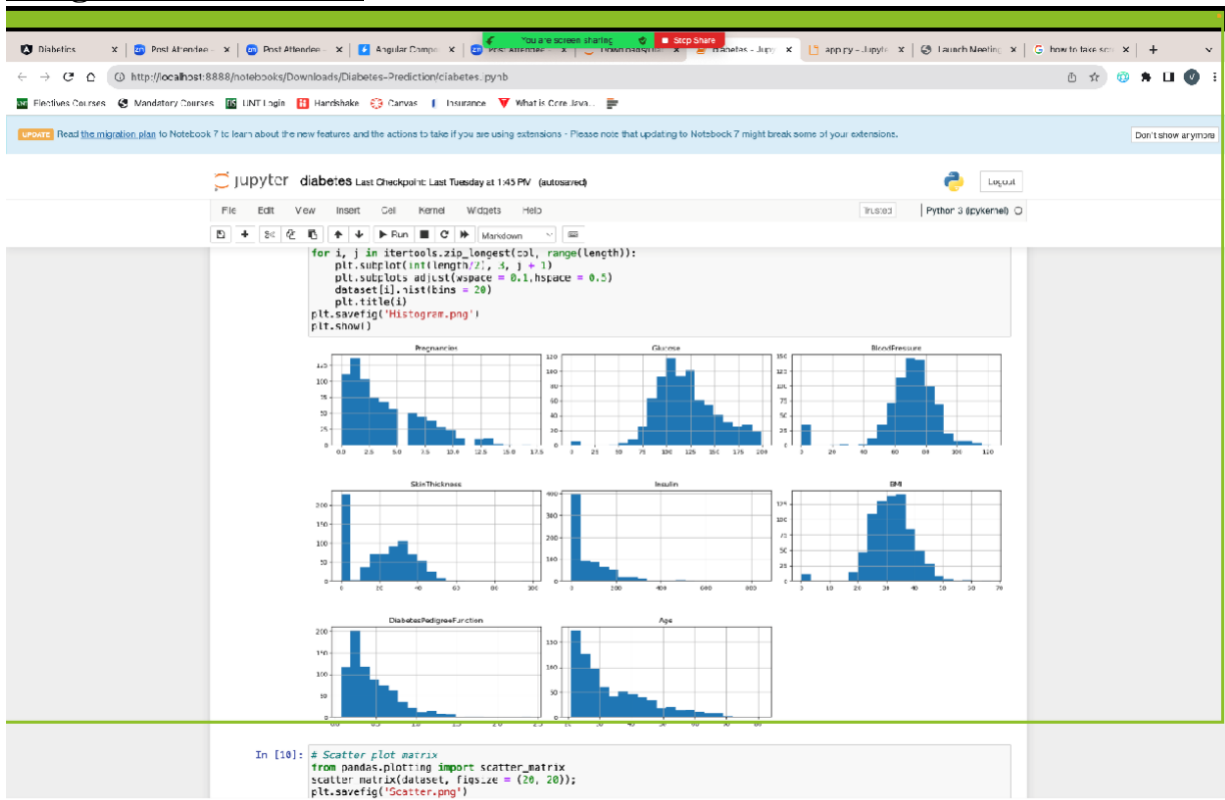
Update Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

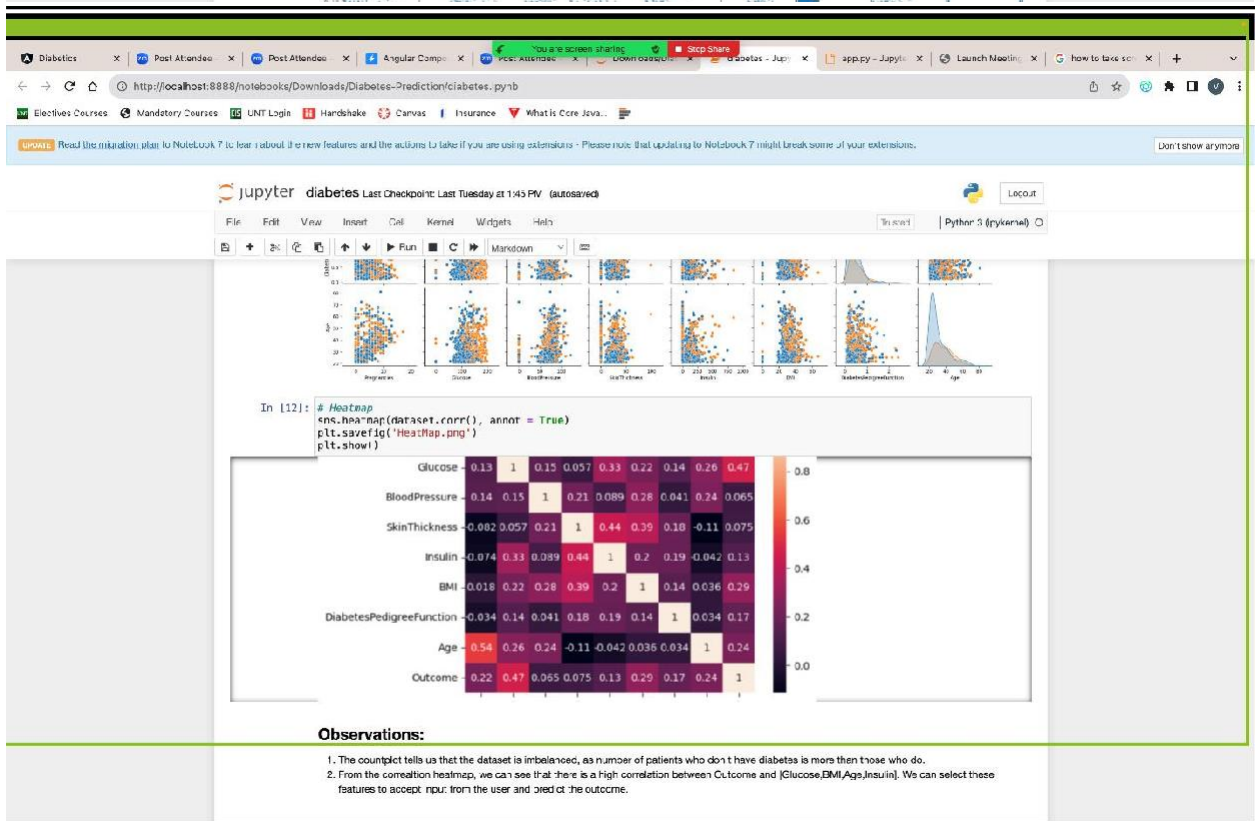
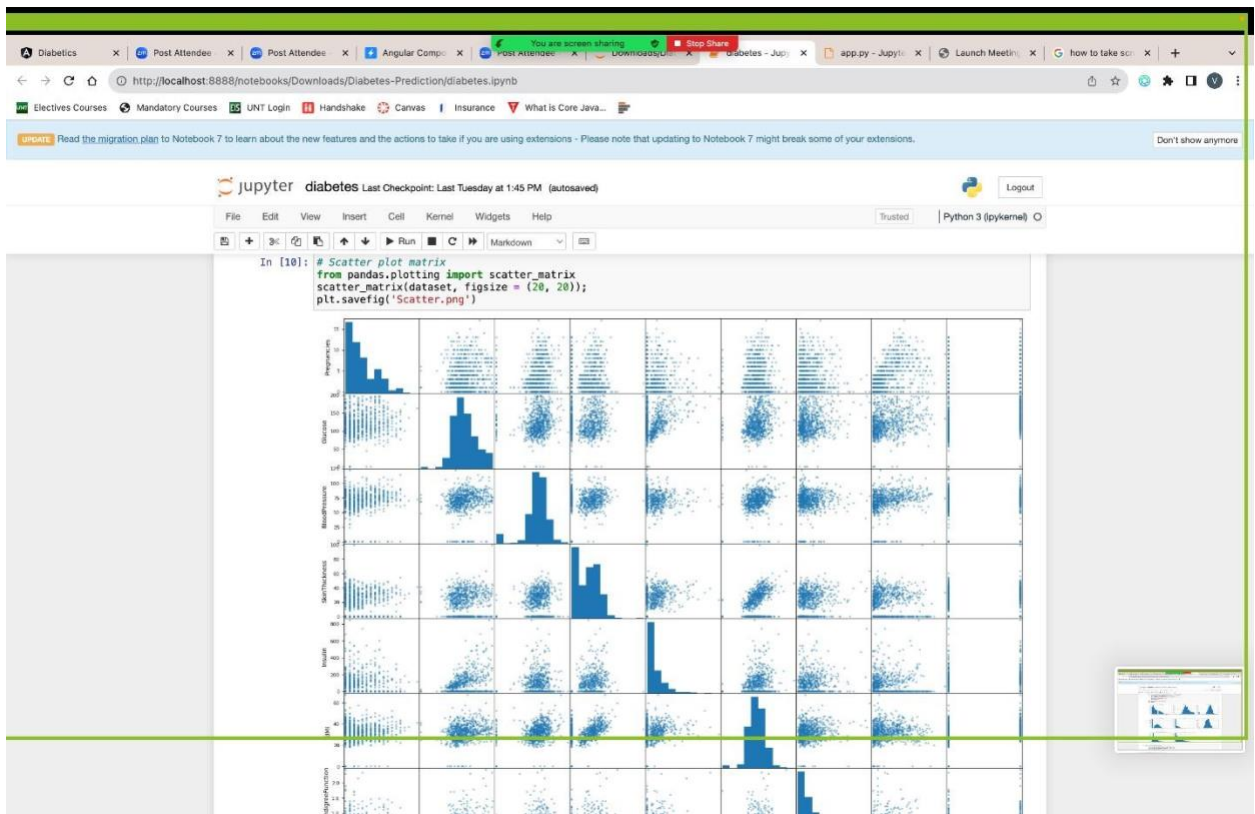
jupyter app.py a day ago Logout

File Edit View Language Python

```
19
20 app = Flask(__name__)
21 CORS(app, origins="*")
22
23 model = load('/Users/vineethreddy/Downloads/Diabetes-Prediction/model.joblib')
24
25 dataset = pd.read_csv('diabetes.csv')
26
27 dataset_X = dataset.iloc[:, [1, 2, 5, 7]].values
28
29 from sklearn.preprocessing import MinMaxScaler
30 sc = MinMaxScaler(feature_range = (0,1))
31 dataset_scaled = sc.fit_transform(dataset_X)
32
33
34 @app.route('/', methods=['GET'])
35 def predict():
36
37
38     # For both GET and POST requests, retrieve parameters from the query string
39     glucose = request.args.get("Glucose")
40     insulin = request.args.get("Insulin")
41     bmi = request.args.get("BMI")
42     age = request.args.get("Age")
43     logging.info(f"Received parameters: Glucose={glucose}, Insulin={insulin}, BMI={bmi}, Age={age}")
44
45     float_features = [glucose, insulin, bmi, age]
46
47     final_features = [np.array(float_features)]
48     prediction = model.predict(sc.transform(final_features))
49
50
51     if prediction == 1:
52         pred = "You have Diabetes, please consult a Doctor."
53     elif prediction == 0:
54         pred = "You don't have Diabetes."
55     output = str(pred)
56     logging.info(f"Final output: {'{}`'.format(output)}")
57
58     return jsonify({'output': output})
59
60
61 if __name__ == "__main__":
62     app.run(debug=True)
63
```

## Images from backend:





```

Last login: Thu Nov 30 18:18:20 on tty005
user@f42323b:~$ command -v curl; command
/usr/bin/curl
user@f42323b:~$ sudo systemctl restart jupyterlab
user@f42323b:~$ systemctl status jupyterlab
● jupyterlab.service
   Loaded: loaded (/etc/systemd/system/jupyterlab.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-11-30 18:18:20 UTC; 1min 1s ago
     Main PID: 1000 (jupyterlab)
       CGroup: /systemd/system/jupyterlab.service
               └─1000 /usr/bin/jupyterlab

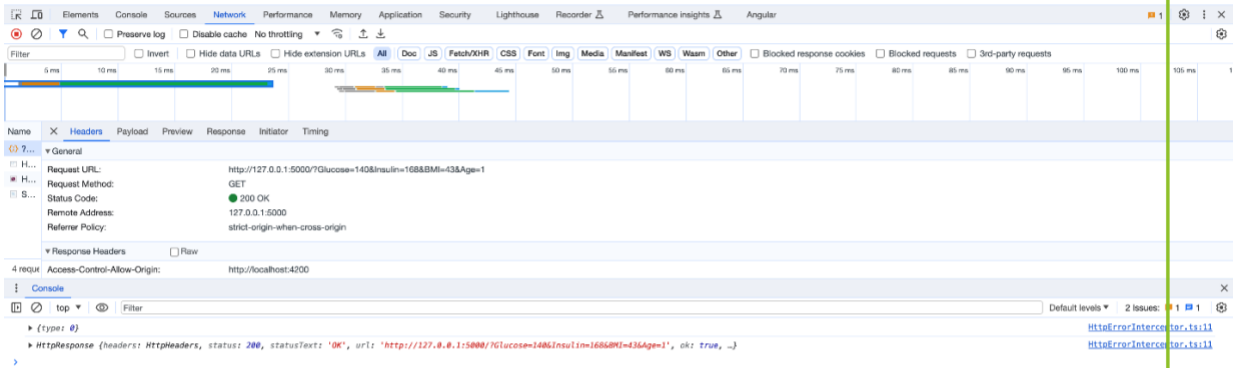
Please see the systemd documentation for more details.

```

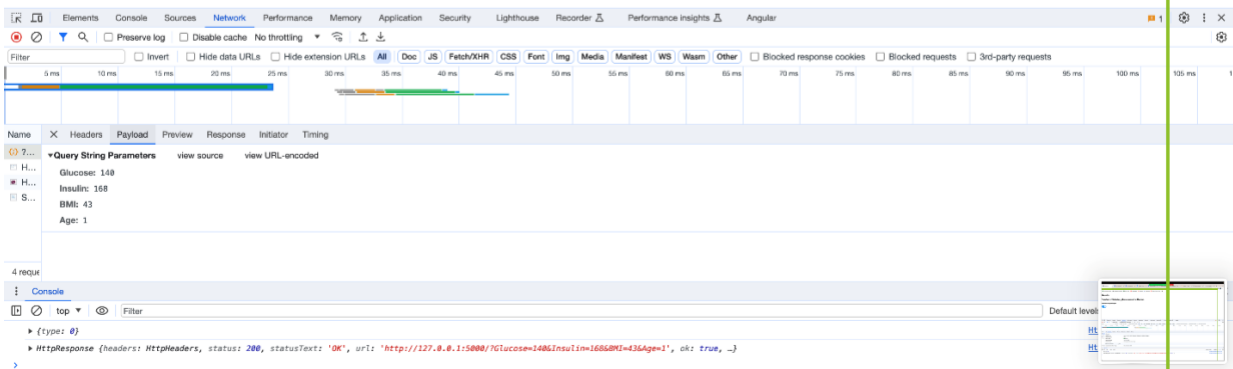
[illegible]

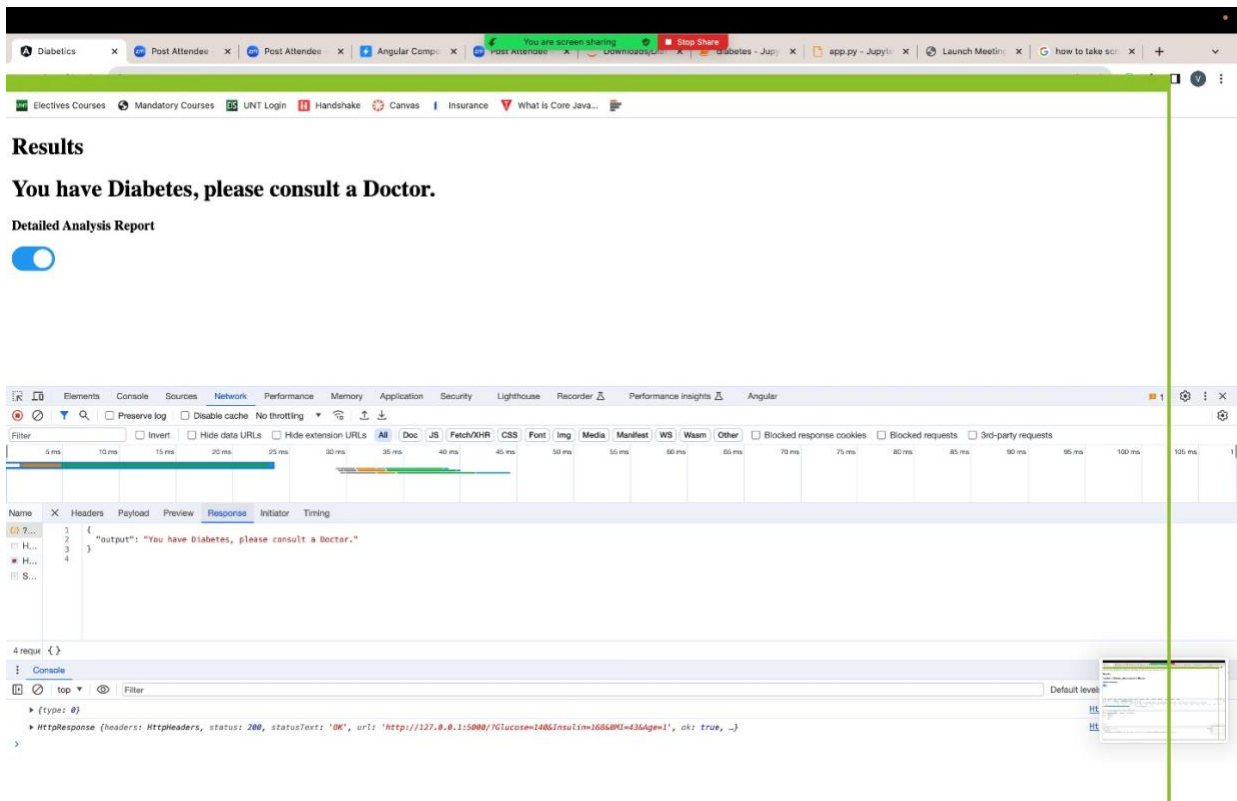
Diabetics x Post Attendees x Post Attendees x Angular Comp x Post Attendees x Download x Diabetics - Jup x app.py - Jup! x Launch Meeti x Google to take sc x + v

## Detailed Analysis Report



## Detailed Analysis Report





These last set of screenshots show how the data is traversed to the frontend which shows the header, payload, response.

Enhancements done are:

- Replaced pickle with joblib as a replacement for pickle to work efficiently on arbitrary Python objects containing large data, in particular large numpy arrays in the current Flask application.
- Transformed from the Basic UI to a UI having fields which having the report analysis.
- Resolved CORPS issue at the backend and front end.



### **3. Task 3: UI Demonstration**

Link to the video demo: [https://myunt-](https://myunt-my.sharepoint.com/:v:/g/personal/suhassiddarajgaritellatakula_my_unt_edu/EbuGht170fNFv-E2KxGKV4sB9Luo87aGUSjs_Vu3CFnJrQ?e=NAQMFI)

[my.sharepoint.com/:v:/g/personal/suhassiddarajgaritellatakula\\_my\\_unt\\_edu/EbuGht170fNFv-E2KxGKV4sB9Luo87aGUSjs\\_Vu3CFnJrQ?e=NAQMFI](https://myunt-my.sharepoint.com/:v:/g/personal/suhassiddarajgaritellatakula_my_unt_edu/EbuGht170fNFv-E2KxGKV4sB9Luo87aGUSjs_Vu3CFnJrQ?e=NAQMFI)