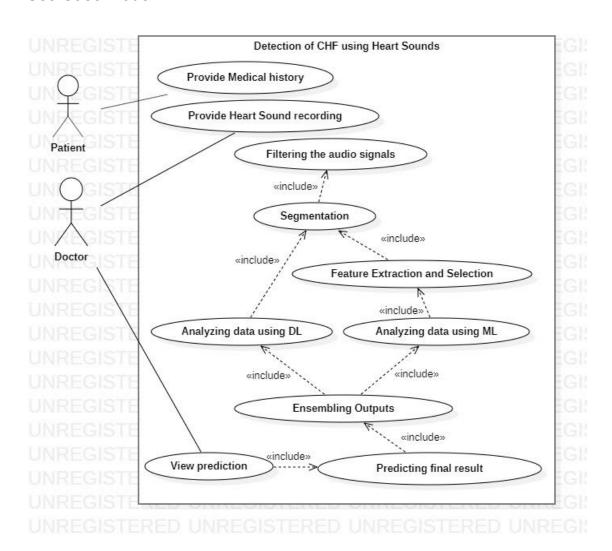
FINAL SUBMISSION GROUP No - 9

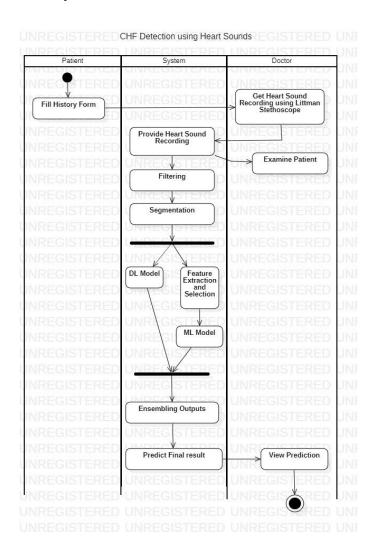
Shreya Kajbaje 7062 Shreya Mahajan 7060 Riya Parekh 7056 Rashmi Mokashi 7021

Assignment 2

Use Case Model -

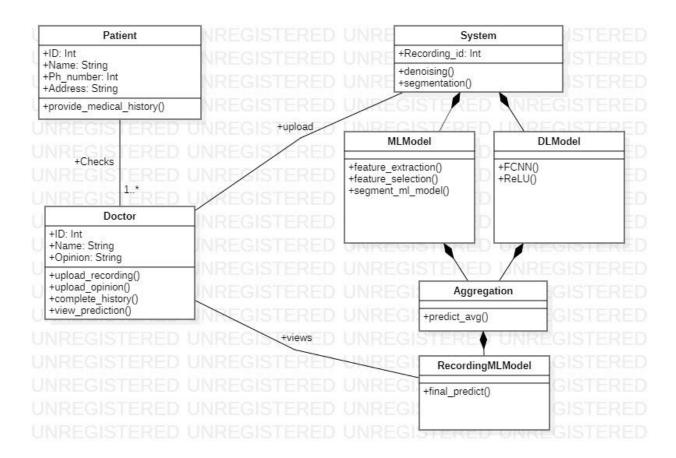


Activity Model -



Assignment 4 & 5

Class Model -



Code -

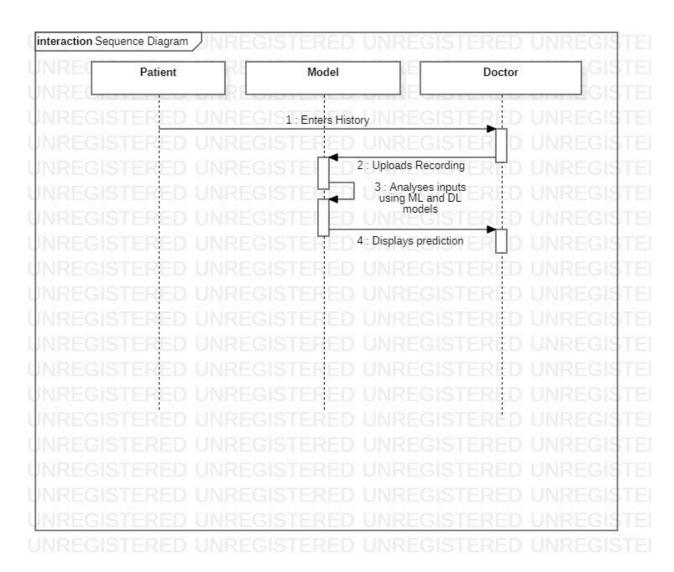
```
#include<iostream>
using namespace std;

class Patient {
   public:
   int ID;
   string Name;
   int ph_number;
   string Address;
   Patient(){
      ID = 404;
      Name = "abc";
   }
}
```

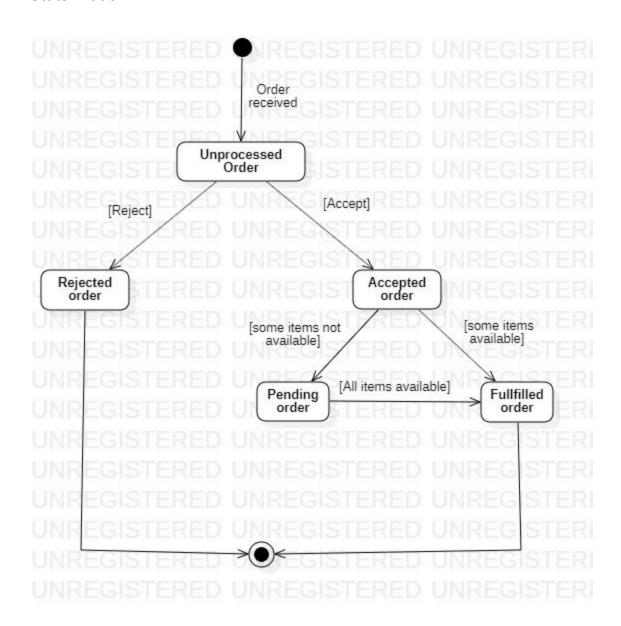
```
ph number = 1234567890;
    Address = "PVGCOET";
    cout<<">>>>>From Patient Class<<<<<"<endl;
    cout<<"ID: "<<ID<<endl;
    cout<<"Name: "<<Name<<endl;
    cout<<"Contact: "<<ph_number<<endl;
    cout<<"Address: "<<Address<<endl;
  }
  void provide_medical_history(){
    cout<<"In provide_medical_history()"<<endl;</pre>
  }
};
class Doctor {
  public:
  int ID;
  string Name;
  string Opinion;
  Patient patient;
  Doctor(){
    ID = 404;
    Name = "abc";
    cout<<endl<<">>>>>>From Doctor Class<<<<<endl;
    cout<<"ID: "<<ID<<endl;
    cout<<"Name: "<<Name<<endl;
  }
  void upload_recording(){
    cout<<"Recordings uploaded"<<endl;
  }
  void upload_opinion(){
    cout<<"Opinion: No"<<endl;
  void complete_history(){
    patient.provide_medical_history();
    cout<<"No BP"<<endl<<"No Diabetes"<<endl;
  }
};
class System{
  public:
  int recording_id;
```

```
void denoising();
  void segmentation();
  System(){
     cout<<endl<<"Recording ID: 101"<<endl;
  }
};
class MLModel: public System{
  void feature_extraction();
  void feature_selection();
  void segment_ml_model();
};
class DLModel: public System{
  void FCNN();
  void ReLU();
};
class Aggregation: public System{
  void predict_avg(MLModel ml, DLModel dl);
};
class RecordingMLModel: public System{
  Aggregation aggr;
  void final_predict(Aggregation aggr);
};
int main(){
  cout<<"Welcome!!"<<endl<
  Doctor d1;
  System s1;
}
```

Sequence Model -



State Model -



Code -

#include<iostream>
#include <stdlib.h>
#include<time.h>
#include<math.h>
#define MAX 2

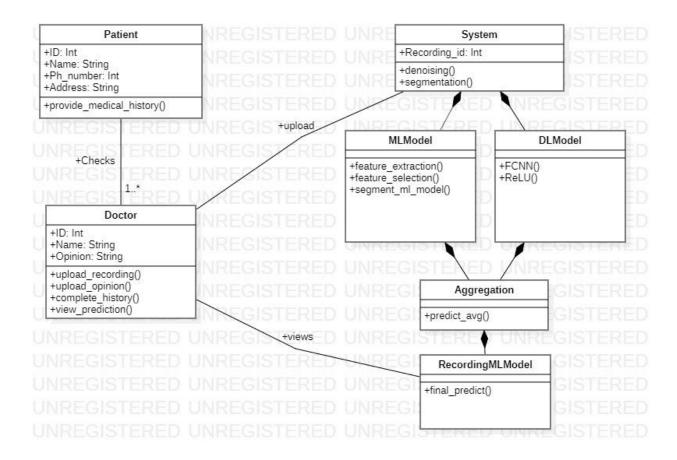
```
using namespace std;
class item{
  public:
   int quantity;
     int order_no;
     int available;
  item(){
     quantity = 3;
     srand(time(NULL));
     available = rand() % 2;
  }
  order(int quan){
     if(quantity>0){
       cout<<"Order received"<<endl;</pre>
       cout<<"Unprocessed Order"<<endl;</pre>
     }
     if(quan<quantity){</pre>
       cout<<"Order Accepted!"<<endl;</pre>
       if(available==0){
             cout<<"Order Pending!"<<endl;</pre>
       while(available==0){
          srand(time(NULL));
          available = rand() % 2;
       }
       if(available==1){
       cout<<"Order Fullfilled!"<<endl;
       }
     else{
       cout<<"Order Rejected!"<<endl;
     cout<<"END STATE!!!"<<endl;
```

}

};

```
int main(){
  cout<<"START STATE"<<endl;
  item i1;
  i1.order(2);
  return 0;
}
OUTPUT:
PS C:\Users\Manisha\Desktop> ./a.exe
START STATE
Order received
Unprocessed Order
Order Fullfilled!
END STATE!!!
PS C:\Users\Manisha\Desktop> ./a.exe
START STATE
Order received
Unprocessed Order
Order Accepted!
Order Fullfilled!
END STATE!!!
PS C:\Users\Manisha\Desktop> ./a.exe
START STATE
Order received
Unprocessed Order
Order Accepted!
Order Fullfilled!
END STATE!!!
PS C:\Users\Manisha\Desktop> ./a.exe
START STATE
Order received
Unprocessed Order
Order Accepted!
Order Pending!
Order Fullfilled!
END STATE!!!
PS C:\Users\Manisha\Desktop>
```

GRASP Pattern -



GOF Pattern -

