Vaccination System

Created By: Kaleb Miller, Robert Silvey, Carter Smith, &
Anthony Telerico

What is it?

CVIS - covid vaccine information system, is a software that will help KSU roll out vaccinations across its campuses in an effective method. CVIS will allow students and faculty to create, view, and manage appointments to help ease to process of getting vaccinated.

Goal:

Create a dynamic system that allows people/staff to efficiently execute the vaccination process

Step 1: Choose an effective language and means of storing data: **Python** and **SQL Database**

The database should include:

- Ability to store patient email and ID, insurance validity, vaccine brand
- Allow users to see their vaccination date, time, campus location, and appointment ID
- Allow an automatic order for more vaccines when low stock
- Ability to track Campus names, vaccines in stock, vaccines given, and revenue generated.

Choosing a Language: Python

Why we chose Python

- Extensive list of libraries to work with
- Easy incorporation of a GUI
- Can incorporate SQL database

Choosing a system: SQL Database

Why we chose an SQL Database

- Simple to design and maintain a schema
- Incorporates well with python
- Easy to manipulate and add data

Python Libraries

TKinter - Used to create our GUI

Datetime - Supplies classes for manipulating dates and times.

Matplotlib - Plotting library for python that creates easy to read graphs for displaying data

Smtplib - defines SMTP client session's that can be used to send mail (emails) out to users

Ssl - Secure Sockets Layer, designed to create secure connection between client and server

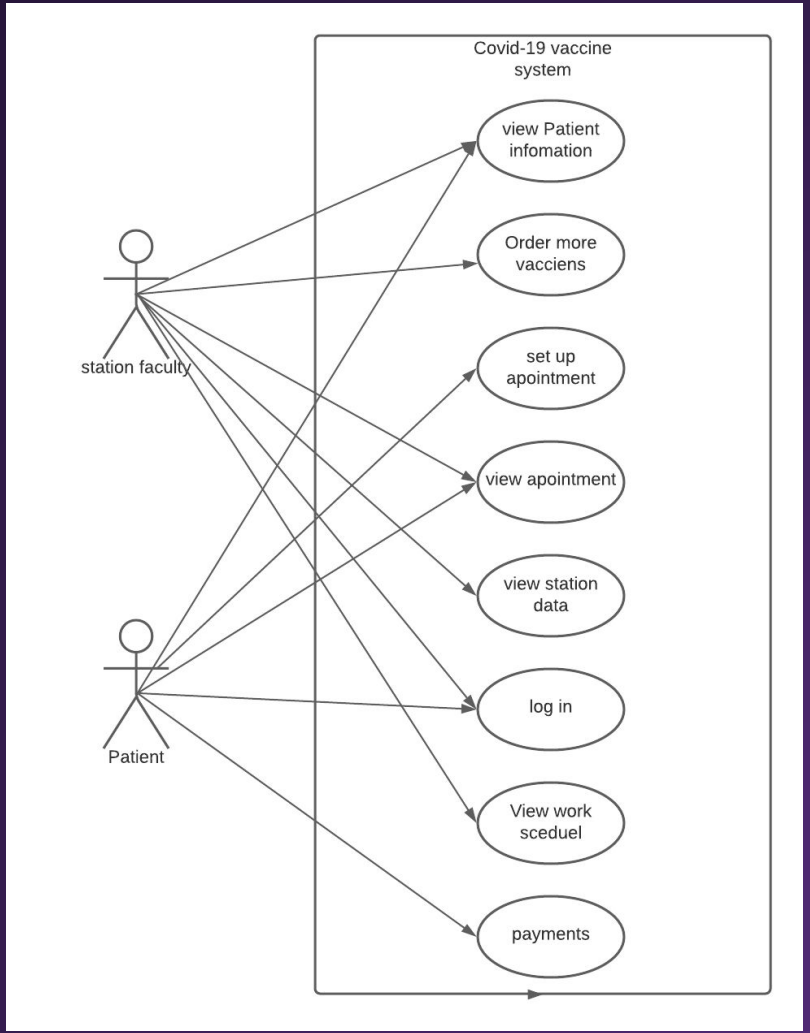
Mysql-connector - Library to connect python code to SQL database

Random - pseudo-random number generator

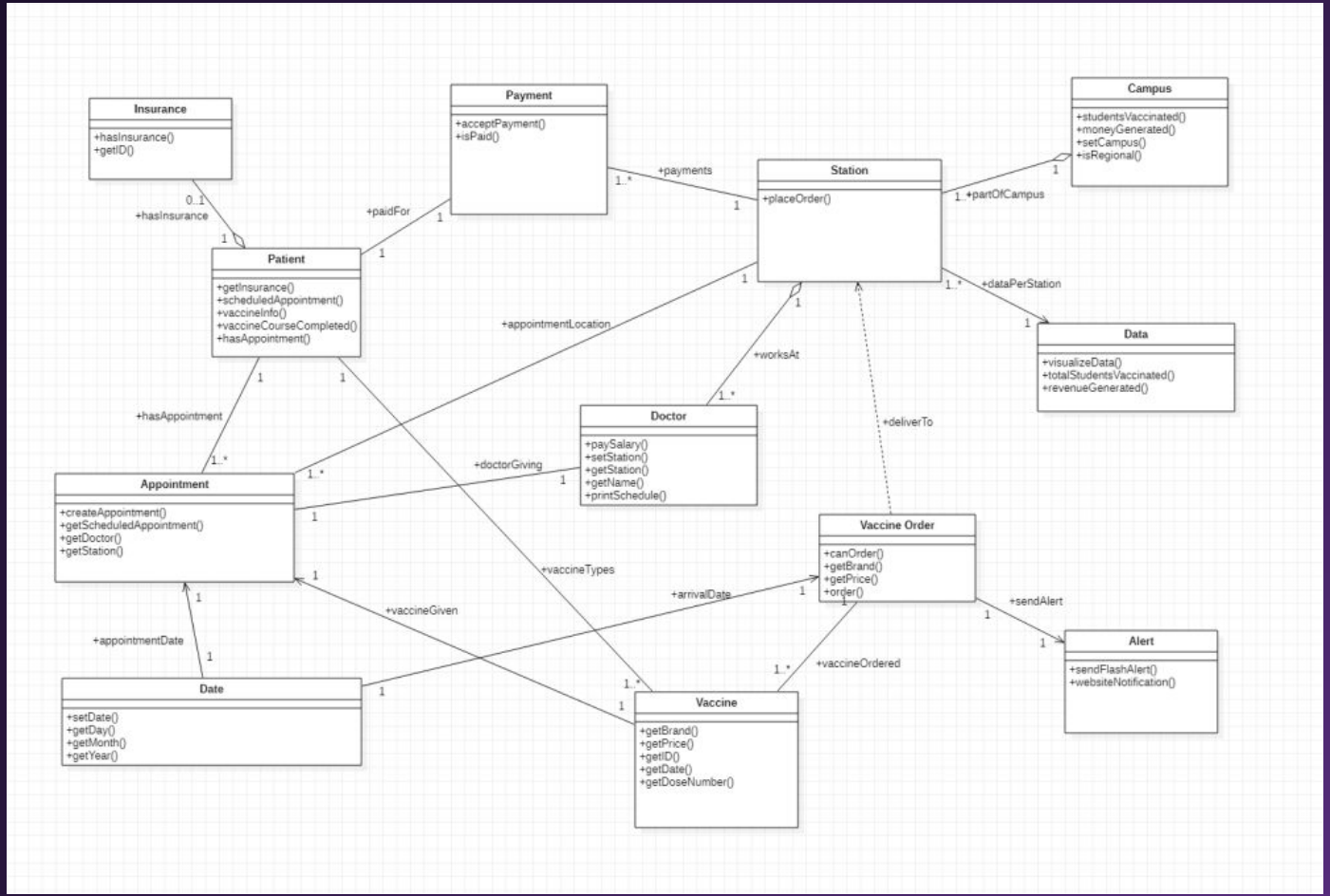
Designing classes and relationships.

- Classes had to be well defined so multiple developers could understand exactly what was going on.
- Class diagram was designed around the idea that the SQL database was the root of our program
- Revised class diagram several times throughout coding process.

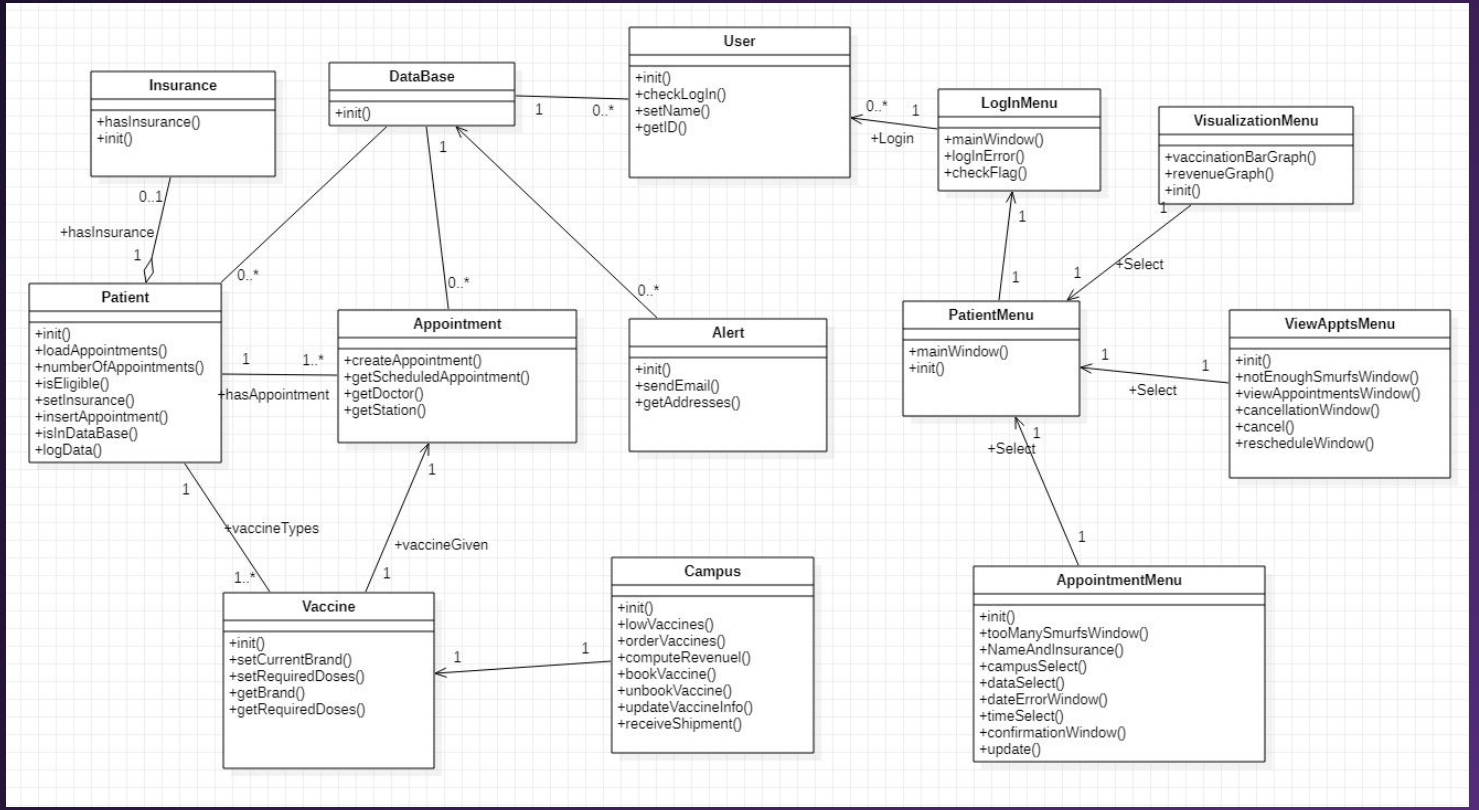
Use Case Diagram



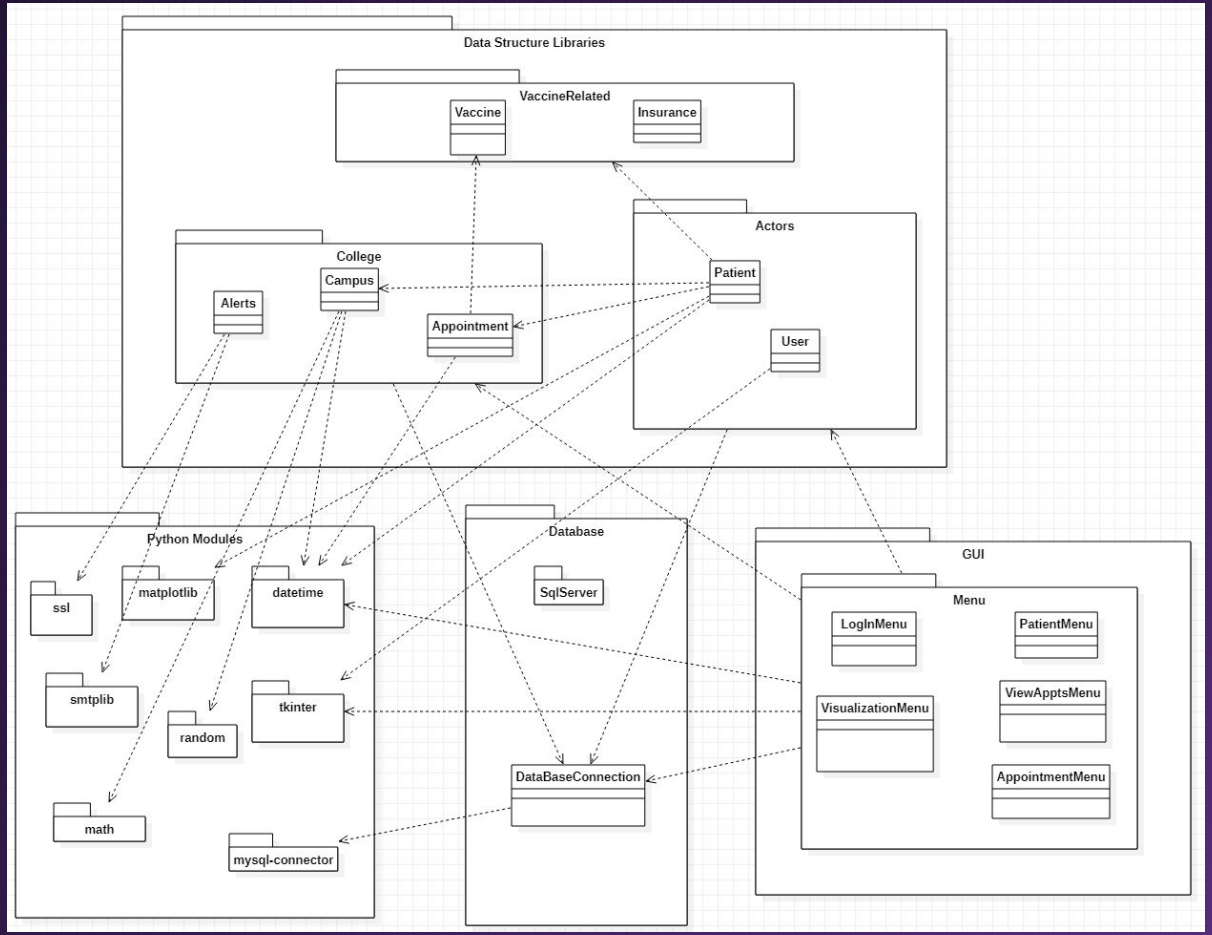
Class Diagram Part 3



Class Diagram Part 4



Class category diagram



Setting up the database

- Setting up SQL database was priority #1. The database is the foundation of the program
- MySQL server was setup and managed by PHPmyAdmin. A free software tool writing in PHP to quickly and easily deploy a database schema
- SQL database is connected to python code by the use of the Mysql-connector library
- The SQL Database is meant to simulate the project as if it were created in Kent FlashLine

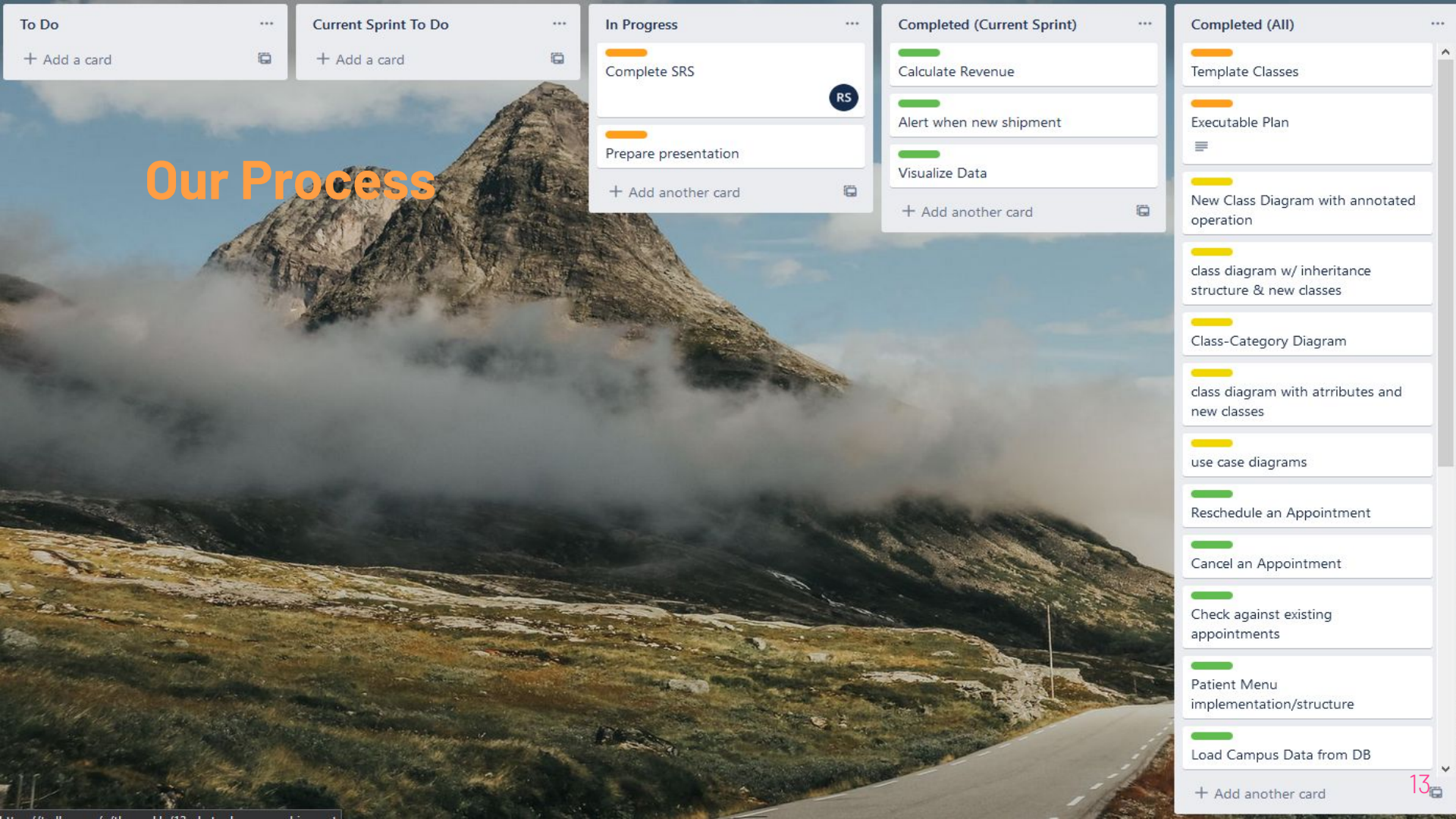
SQL database example

Example of the appointment database that stores information regarding the upcoming or past appointment.

```
CREATE TABLE `appointment` (
  `AppointmentID` int(11) NOT
NULL,
  `UserID` int(11) NOT NULL,
  `Campus` varchar(15) NOT NULL,
  `AppointmentDate` date NOT
NULL,
  `AppointmentTime` time NOT
NULL,
  `VaccineBrand` varchar(45) NOT
NULL,
  `Complete` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4;
```

AppointmentID	UserID	Campus	AppointmentDate	AppointmentTime	VaccineBrand	Complete
1051	1110	Stark	2021-05-03	09:51:00	Johnson&Johnson	0
1420	1252	Kent	2021-05-06	10:35:00	Pfizer	0
1951	1330	Salem	2021-05-13	11:00:00	Pfizer	1
1429	1131	East Liverpoo	2021-05-20	09:30:00	Moderna	0
2142	1800	Trumbull	2021-05-08	06:45:00	Moderna	0

Our Process



To Do

+ Add a card

Current Sprint To Do

+ Add a card

In Progress

- Complete SRS (RS)
- Prepare presentation

+ Add another card

Completed (Current Sprint)

- Calculate Revenue
- Alert when new shipment
- Visualize Data

+ Add another card

Completed (All)

- Template Classes
- Executable Plan
- New Class Diagram with annotated operation
- class diagram w/ inheritance structure & new classes
- Class-Category Diagram
- class diagram with atrributes and new classes
- use case diagrams
- Reschedule an Appointment
- Cancel an Appointment
- Check against existing appointments
- Patient Menu implementation/structure
- Load Campus Data from DB

+ Add another card

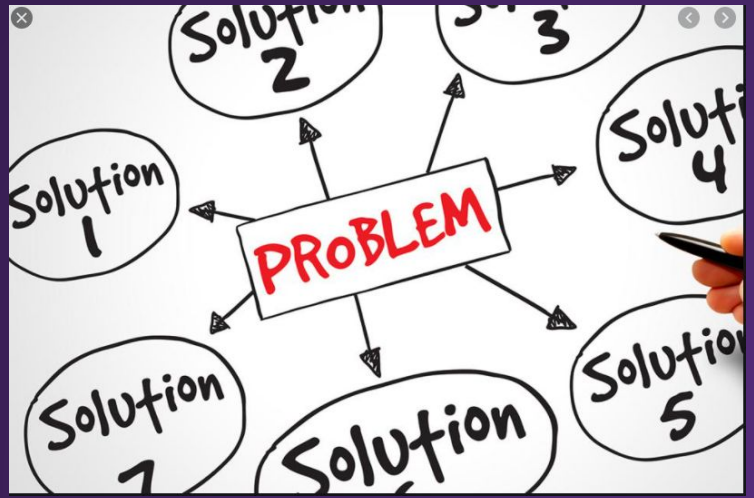
Implementing the GUI

```
class ViewApptsMenu:
    def __init__(self, patient):
        self.currentPatient = patient
        self.selection = ""

    def notEnoughSmurfsWindow(self):
        if self.currentPatient.numberofAppointments() == 0:
            self.root = tk.Tk()
            self.root.geometry('400x100')
            label = tk.Label(self.root, text="Current User has no scheduled appointments")
            accept = tk.Button(self.root, text ="OK", width = 20, command = lambda:[self.root.destroy()])
            label.pack()
            accept.pack()
            self.root.mainloop()
        else:
            self.viewAppointmentsWindow()

    def viewAppointmentsWindow(self):
        self.root = tk.Tk()
        self.root.title("Select Appointment")
        self.root.geometry('400x150')
```

TKinter - was the primary library used when implementing the GUI.

Problems faced

- Originally was a c++ program
- Setting up collaboration program (github) to allow developers to work on project simultaneously
- Developers having a variety of experience with the languages used.
- Time, any software can be improved with more time



Room for Improvement?

- Alert email send available dates
- Creating Dedicated Server for SQL database.
- Adding a more diverse user category to the system such as Employee's and admins.
- Better menu navigation


We will now do a live DEMO of our software.
Enjoy!