Chawon White
CIS441 Midterm Write-up
"Seven Deadly Cins"

The sin I suffered was pride. In the beginning, our group formed and idea for a drink app that sounded like fun and an easy task. We were naïve. We were going to put function beyond function in our app, and everything was cool… until we got to the graphical user interface. Once we got here, everything we made went to shit; error after error, glitch after glitch, and a programming language NO ONE was ready for cut us down.

We started off well. Everyone in the group knew how to code in C++, so it seemed like a natural choice to code in. We would all code our individual pieces, bring them together, and make sure everything played nice. The first half, there wasn't anything code-wise I needed to learn; I may have missed some steps, but coding the back end was fairly straightforward. I had my piece done and was helping the others catch up. I tried helping Nathan fix his Iterator class, but I think he picked one of the more complicated classes we could have used and I was in over my head. Nevertheless, I tried to help him as best as I could, until he decided to scrap the class to work on the Search class.

Something I learned a while back that was reinforced in this class was that people do not code the same. For example, when I code using C++, the top of my files look kind of like this:

```
#include <iostream>
#include <string>
using std::cout;
using std::endl;
using std::string;
```

While the others might look like this:

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

This wasn't a new concept I learned, this was something I noticed years ago when I first figured out that the namespace std has a lot of stuff in it, and you can just grab what you need. This may seem like an irrelevant fact, but it acted kind of like a precursor to how things would go. Because we are a team of people who code differently, we had some trouble early on linking the classes together. Most of the problems we ran into for the backend took Antowyn a while to solve; he did a lot of the work on the backend of the program, making sure that any class that interacted with the drink class worked properly. While he did this, I broke from the group and started researching into Qt, a language I've never heard of. On the outside, it reminded me of Java or C#. I thought it would be fairly easy to pick up and implement for our GUI.

Qt has been nothing short of nightmarish since the beginning. What I thought could be a skill easily transferrable from Java and C# was something so similar and so different. I'd say it's like falling asleep in a waterbed; sounds easy enough, but if you make a wrong move you could destroy everything. I read some tutorials, watched some videos, and still ran into problems with the GUI. Once I got one thing working, everything else stopped working. I can admit I bricked the GUI in almost every iteration we have, except for v5. I tried to figure it out on my own, which may have been the problem; I believed I could do it with little help because the style seemed reminiscent of other things I've worked on.

Chawon White
CIS441 Midterm Write-up
"Seven Deadly Cins"

I developed the first five versions of the GUI for our app. The reason why there were five versions of our GUI is because Qt was very difficult to learn and implement at the same time. The first version was just a design of what we wanted the app to look like, and every version after it until v5 was an attempt to get it to work with the backend. The fifth version has five main use cases, all tied to the buttons on the home screen. I tried linking the "Surprise Me!" or the randomDrink button to an action. Here is what I did:

```
void sevencins::on_randomDrinkButton_clicked()
{
    qsrand(0);
    int randomDrink = qrand() % 21 + 1;
    Cocktail toOutput = drinks.at(randomDrink);
    this->setWindowTitle(QString::fromStdString("7DC - " +
toOutput.getName()));
    this->drinkName->setText(QString::fromStdString(toOutput.getName()));
    this->drinkInfo-
>setText(QString::fromStdString(toOutput.getDescription()));
    Recipe temp = toOutput.returnRecipe();
    for(int i = 0; i > toOutput.RecipeLength(); i++)
    {
       this->drinkInfo->append(QString::fromStdString("Step " + (i+1) +
temp.returnRecipe().at(i) + "\n"));

    }
}
```

What is supposed to happen here is as follows: seed the random number generator, get a random number 1-21, get the drink at the map index for the random number, then print out the information for the drink and change the picture. Since there are errors around what I am trying to do, the next thing I want to try is to fix what I broke, one piece at a time. I think if we had more time, we might be able to fix it as a team and turn in something awesome.

Throughout the process, one thing I've learned is that a narrow focus will bind you to trouble. I thought that since we were working on making the backend play nice, the GUI would come in time. This was not the case; I was not prepared for how difficult it would be to code a GUI that would work with the backend flawlessly. If we started coding the GUI in the first couple of sprints, I think we would have had less trouble down the line. I have a new respect for developers everywhere; if you make a mistake as a dev, no one forgets it. Now, after trying to develop a piece of software, I can see why developers get into trouble and have to scrap an entire project.

Some links I used; there were more, but they would have taken up too much space:

https://doc.qt.io/qt-5/designer-using-a-ui-file.html

https://forum.qt.io/topic/61666/qt-random-number

https://doc.qt.io/qt-5/qtwidgets-widgets-calculator-example.html