Team: Financial Modeling Girls
By: Natalie To, Mina Yuan, Haley Zhang

Our Simple but Great Idea: Eliminating time searching for available tables/cubicles at libraries.

The Problem

Being students our team can relate to the struggles of having to find a quiet and available study space. Many students on campus prefer to go to libraries where there are quiet zones provided, however when exams kick in people spend days at the library. Tables become crowded and people start to search every floor to find one available seat, but this takes time. Especially, when students study at libraries for days the amount of time is accumulated and could have been used for studying and further preparing for exams. Usually, there isn't a lot of time to study for multiples consecutive finals so any extra time a student could gain is very beneficial.

Execution Plan

We plan to create a system in libraries where students can login their student ID whenever they are seated at a table. Making this program in real time, it continually takes those logins to signal that a specific seat in the library is occupied. When an individual leaves, they must sign out so that the program can know when to record that the seat is now available. There is no time limit as to how long a person can spend at this space, but the program is also able to book these study spaces. This concept of booking cubicles will create an incentive for students to use our program because it would prevent them from being kicked out of their space. Our program is similar to a room booking system, but instead implemented at tables and cubicles. We want to effectively minimize the time wasted searching for study spaces.

Side Note:

Unfortunately, as beginner programmers with little to none experience we were unable to create a working and complete program for our idea. We played around with some programs and googled some codes that would be helpful in creating our system and this is as far as we got.

Program:

Below is a sample code on R from https://www.r-bloggers.com/simple-user-interface-in-r-to-get-login-details/ where we hope to have a login prompt the user for their student ID and password. This will be what the user will see initially when seating at an empty table. You can plug this code into R.

```
require(tcltk)
tt <- tktoplevel()
tkwm.title(tt, "Get login details")
Name <- tclVar("Login ID")
```

```
  Password <- tclVar("Password")
  entry.Name <- tkentry(tt,width="20", textvariable=Name)
  entry.Password <- tkentry(tt, width="20", show="*",
                            textvariable=Password)
  tkgrid(tklabel(tt, text="Please enter your login details."))
  tkgrid(entry.Name)
  tkgrid(entry.Password)

  OnOK <- function()
  {
    tkdestroy(tt)
  }
  OK.but <-tkbutton(tt,text=" OK ", command=OnOK)
  tkbind(entry.Password, "<Return>", OnOK)
  tkgrid(OK.but)
  tkfocus(tt)
  tkwait.window(tt)

  invisible(c(loginID=tclvalue(Name), password=tclvalue(Password)))
 }
 credentials <- getLoginDetails()
 ## Do what needs to be done
 ## Delete credentials
 rm(credentials)
```

This next part is the body and the background for our system using C++

file student: student id database (each line contains 1 student number, integer)
file library:
dataframe: columns: spot number-char, availability-int [0,1], floor-int [-1,1,2,3,4,5], type-char [‘table’, ‘cubicle’, ‘room’])

```
#include <iostream>
```

#include <fstream>

algorithm:

Function 1(temp_info): check if student is in the system

Function 2(floor,type=single): a map to all available tables:

display all available tables
ask user for input: floor, type (default=single, optional)
Case loop:
case floor = “1st” & type = “single”:
display all available tables output specific to that floor
break;
case ...
15 case, 5 floors, three types: cubicle, table, room

Function 3: check in, exit:

ask user for input of their information (when arrive at the table and leave the table)
(or tab the student card if there’s a sensor installed on each table, for check in and out the tables)

Main:

cout>> ask user for input
student_num << cin<< user’s input of their info
Function 1: validate(temp_info)
Function 2: avail_spot
Function 3: checkin (char Username, char PIN)

Sample code:

```
#include <iostream>
#include <fstream>

# Function 1
validate (int student_num)
{
    t=student_num
    infile1.open(file_student) # read it in as a matrix, with 2nd column: 0(empty) or 1(occupied)
    while ( !file_student.eof() )
    {
        getline(file,t);
        lines.push_back( t );
    }
    string first_line = lines[0];
```

```
        if ( first_line.find('1')!=string::npos )
                cout << “login successful”;
        else
                cout << “invalid student number”;
}


# Function 2
void avail_spot{
infile2.open(file_library)
Data_selected = infile2[,2] == 0   //display all available tables in the library
count>>data_selected

cout>> “Enter floor(-1,1,2,3,4,5)”;
floor << cin;
Cout >> “Enter type  ( table, cubicle, room)”;
type << cin;

switch ( floor, type)
case 1: floor ==1 & type == “single”
         if ( available table == FALSE)
            cout<< “ there is no single table in this floor”;
        else
           (statements) // display all available table of the floor and type selected and a graphic of map of
the floor
        break;

case 2: .
         break;
case # : ... // 15 cases= 3 types*5 floors in total...


}

# Function 3
checkin (char Username, char PIN )
{
std::cout << “Enter your Username, then hit the ENTER”<<std::endl;
std::cin >> Username;
std::cout << “Enter your PIN, then hit the ENTER”<<std::endl;
std::cin >> PIN;
std::cout << “Log in completed, you can start using the table ”<<std::endl;
}
```

Future Goals:

With more time we hope to achieve a fully functional program that can be implement in many libraries. Possibly, our program could be on an app students can download, where they can select a specific space they have found available in the library and input their id on the app's interface. This way libraries don't need to make costly purchase to put a build in login system at each table.