GRADR

A Quick Recap

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

Student list may be imported from Excel spreadsheets.

Files stored and organized automatically.

Security

Student data and grades stored on customer’s secure server.

Username and password login.

Scalability

Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

Files stored and organized automatically.

Security

Student data and grades stored on customer’s secure server.

Username and password login.

Scalability

Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

✔ Files stored and organized automatically.

Security

Student data and grades stored on customer's secure server.

Username and password login.

Scalability

Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

✔ Files stored and organized automatically.

Security

✔ Student data and grades stored on customer's secure server.

Username and password login.

Scalability

Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

✔️ Files stored and organized automatically.

Security

✔️ Student data and grades stored on customer's secure server.

⚠️ Username and password login.

Scalability

Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

✔ Files stored and organized automatically.

Security

✔ Student data and grades stored on customer's secure server.

⚠ Username and password login.

Scalability

✔ Client may access data from server with internet access.

Potential for web and app integration in the future.

A Quick Recap

Current Environment

Client currently uses Excel spreadsheets stored locally on her Windows computer.

Manually enter students and assignments for each grade. Manually save and organize files.

Improvements

Automation

❌ Student list may be imported from Excel spreadsheets.

✔️ Files stored and organized automatically.

Security

✔️ Student data and grades stored on customer's secure server.

⚠️ Username and password login.

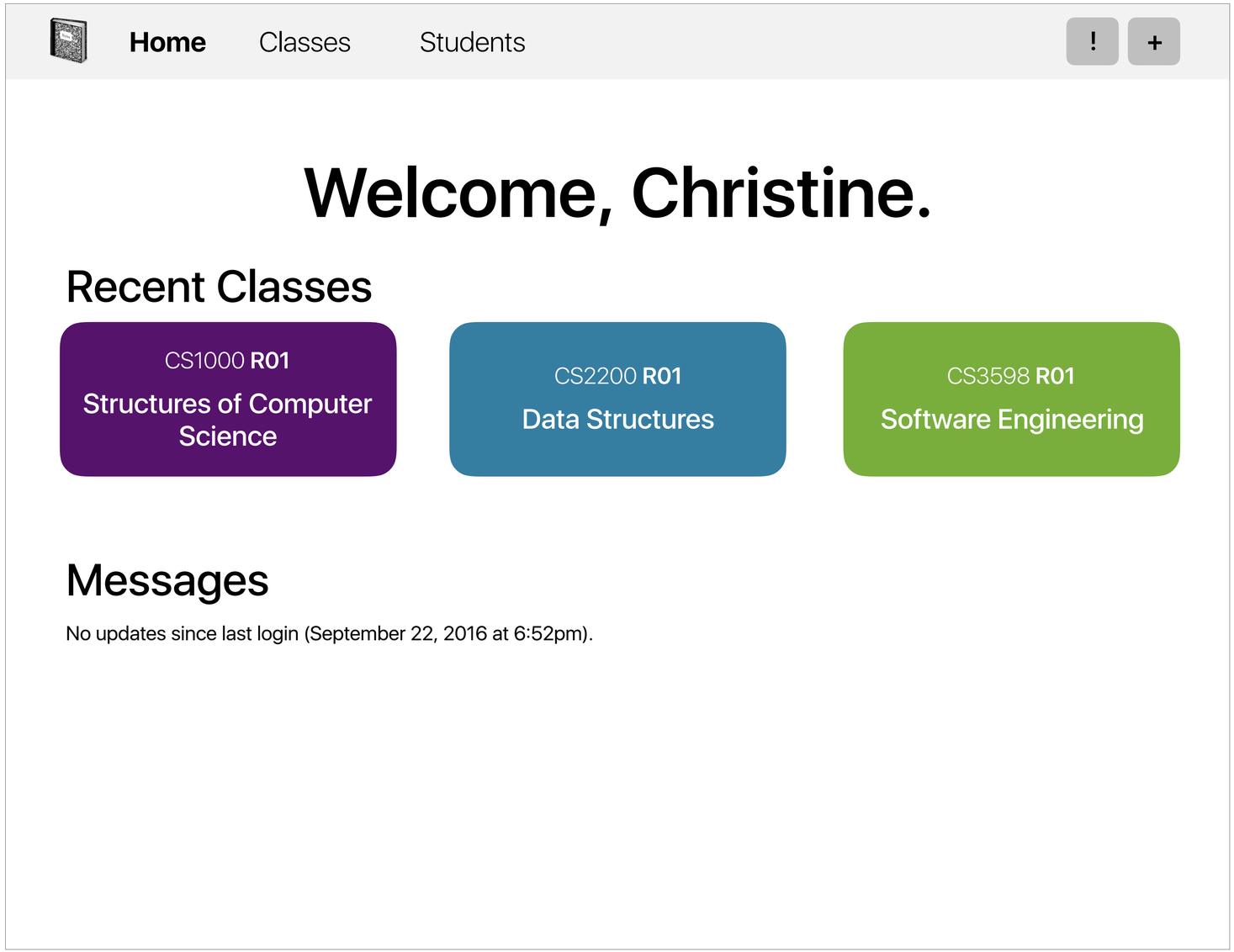
Scalability

✔️ Client may access data from server with internet access.

⚠️ Potential for web and app integration in the future.

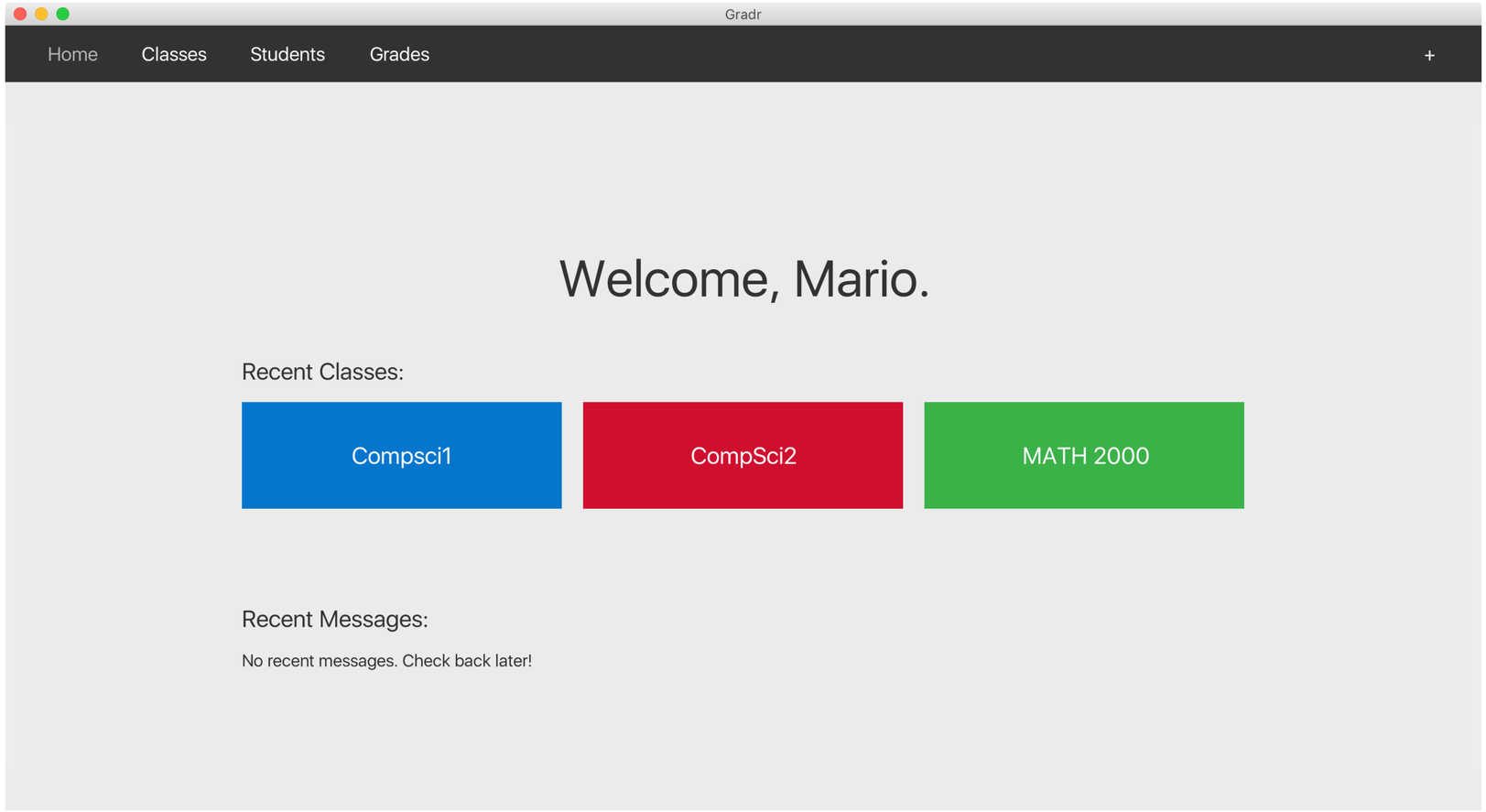
A Quick Recap

GUI prototype



A Quick Recap

Final GUI

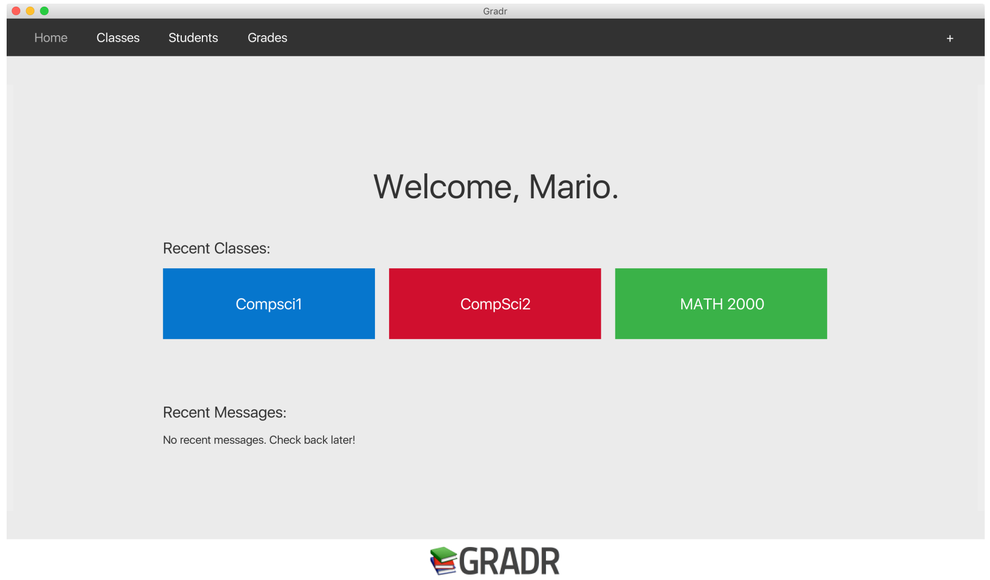


GRADR

Overview

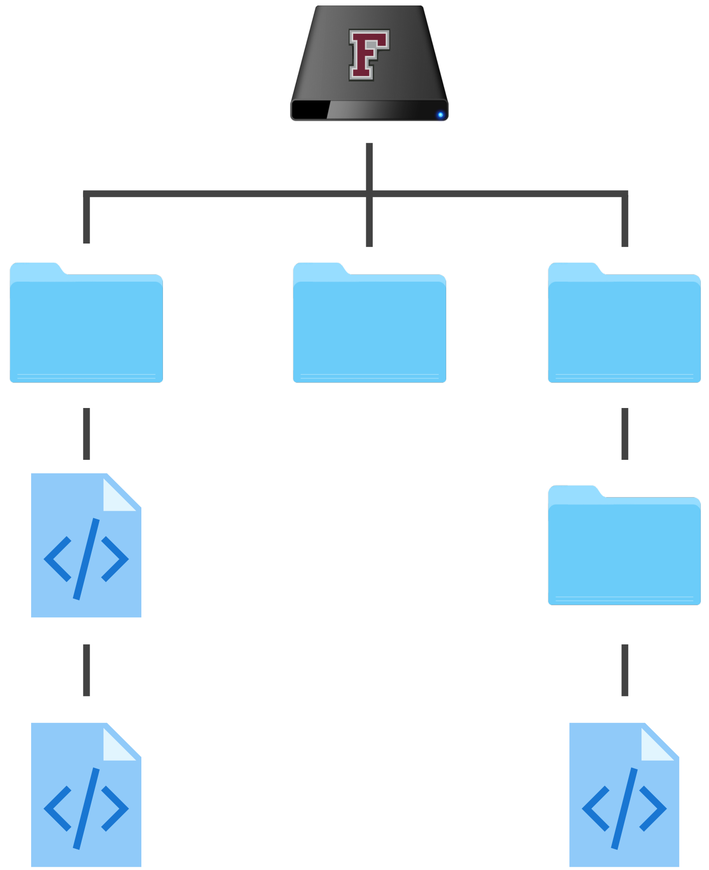
Overview

GUI



Logic

File I/O



File System

File System

Structure

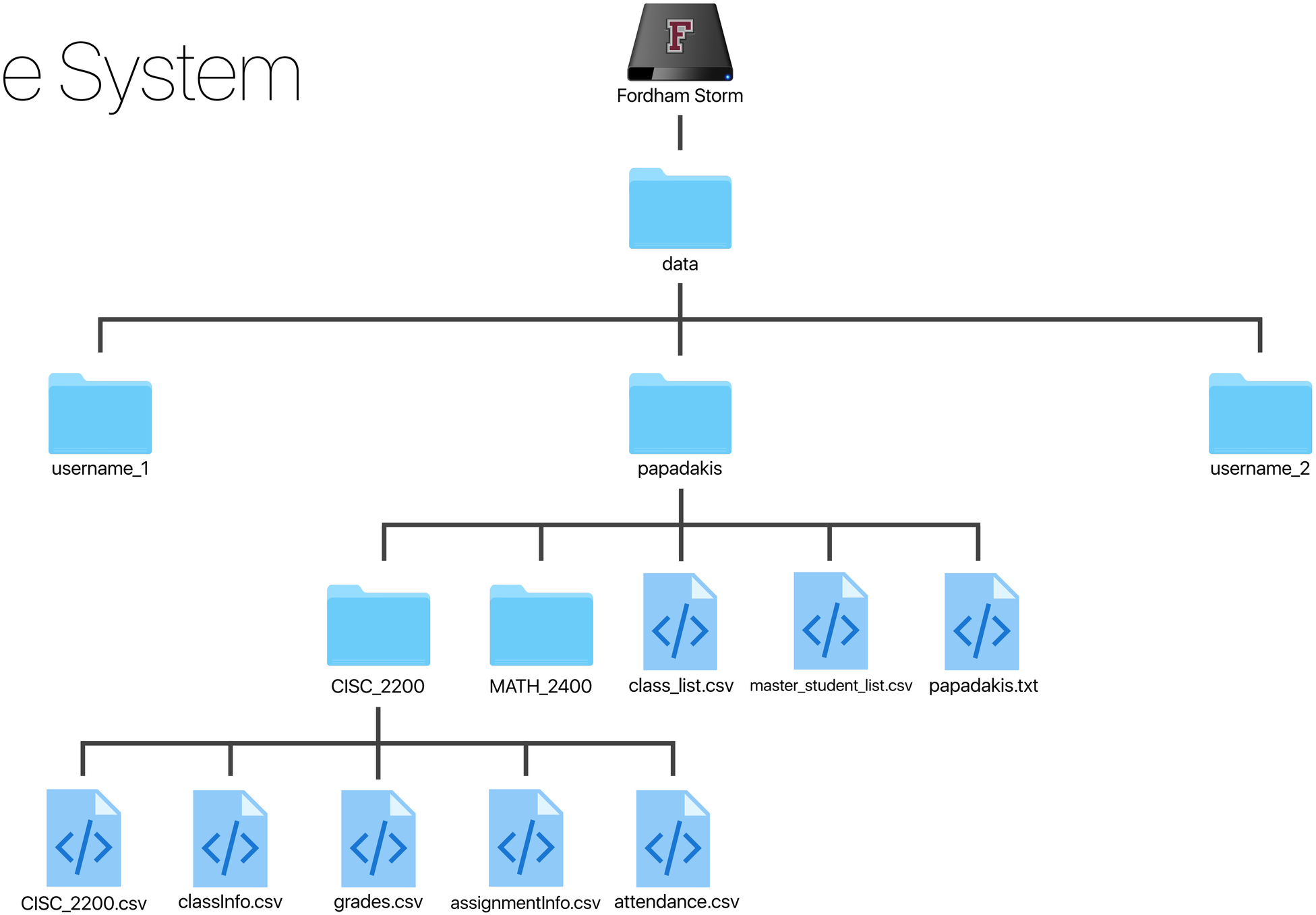
- Main “data” directory
- Directory for each teacher/user, created during registration
  - File containing username and password
  - File specifying which classes to which the user has access
- Directory for each class created
  - Attendance
  - Class information
  - Grades
  - classname.csv (containing list of information from student objects)

File System

```
[brookecantwell@storm Mario]$ pwd
/home/students/FALL15/brookecantwell/CISC3598/gradebook_final/data/Mario
[brookecantwell@storm Mario]$ ls
class_list.csv  Compsci1  CompSci2  Mario.txt  master_student_list.csv
[brookecantwell@storm Mario]$ cat Mario.txt
Mario
pass
Mario
Merendino
[brookecantwell@storm Mario]$ cat class_list.csv
Compsci1
CompSci2
[brookecantwell@storm Mario]$ cat master_student_list.csv
A123,Merendino,Mario,20
A345,Cantwell,Brooke,30
A678,Merendino,Joseph,12
A010321,Subject,Test,30
A9648963,Student,Newest,532
27,Cantwell,Brooke,23
28,Cantwell,Brooke,23
[brookecantwell@storm Mario]$
```

An example of the file system created for each teacher during registration.

File System


Fordham Storm
data
username_1
papadakis
username_2
CISC_2200
MATH_2400
class_list.csv
master_student_list.csv
papadakis.txt
CISC_2200.csv
classInfo.csv
grades.csv
assignmentInfo.csv
attendance.csv

File Input & Output

FileIO class

Methods for reading in and writing to files are **not** contained in the classes on which they operate.

They are aggregated into the File I/O class.

Methods are called on a fileIO object, passing the objects they operate on as arguments.

The File I/O class is composed of methods that accept objects and strings to perform input and output to the different files in the system.

```
246     // update class information when new class is selected from dropdown
247     private void update(String newClassName){
248         FileIO io = new FileIO();
249         className.setText(newClassName);
250         classroom c = new classroom();
251         c = io.getClassInfo(newClassName);
252         className.setText(c.className);
253         section.setText("Section: " +c.section);
254         semester.setText("Semester: " + c.semester);
255         meetingTime.setText("Meeting Time: " + c.meetingTime);
256         location.setText("Location " + c.location);
257         crn.setText("CRN: " + c.crn);
```

Logic

Logic

Major classes

The logic behind our program is built with several classes representing different objects required for the gradebook:

Classes (or classrooms)

Assignments

Students

Teachers

File input/output

Logic

File output

addToMasterStudentList( ): Verify unique student IDs and write to the master student list.

```
239    public static void addToMasterStudentList(student s) {
240        String fileName = (System.getProperty("user.dir") + "/data/" + currentTeacher.username + "/master_student_list.csv");
241
242        boolean newStudent = true;
243
244        try {
245            File outputFile = new File(fileName);
246            FileWriter fw = new FileWriter(outputFile.getAbsoluteFile( ), true);
247            BufferedWriter bw = new BufferedWriter(fw);
248
249            if(!outputFile.exists( )){
250                outputFile.createNewFile( );
251                bw.write("IdNumber,First Name,Last Name, Age");
252                bw.newLine();
253            }
254
255            List<String> tempList = new ArrayList<String>( );
256
257            tempList = readMasterStudentList(currentTeacher.username);
258
259            String search = s.idNumber;
260
261            for (String str:tempList) {
262                if(str.trim( ).contains(search))
263                    newStudent = false;
264            }
265
266            if (newStudent == true) {
267                bw.write(s.idNumber + ',' + s.lastName + ',' + s.firstName + ',' + s.age);
268                bw.newLine();
269                bw.close( );
270            }
271            else
272                System.out.println("Not a new student");
273
274        }
275        catch (IOException ex) {
276            System.out.println(ex);
277        }
278    }
```

Logic

File output

appendStudent( ):
Add a student to the specified class file.

```
193
194     //APPEND STUDENT, CALLED FROM ADD STUDENT
195     public static void appendStudent(student s, String className) {
196         String fileName = (System.getProperty("user.dir") + "/data/" + currentTeacher.username +'/' + className + '/' + className +  ".csv");
197         String gradesFileName = (System.getProperty("user.dir") + "/data/" + currentTeacher.username +'/' + className + '/' + "grades.csv");
198 
199         try {
200             File outputFile = new File(fileName);
201             FileWriter fw = new FileWriter(outputFile.getAbsoluteFile( ), true);
202             BufferedWriter bw = new BufferedWriter(fw);
203 
204             File gradesOutputFile = new File(gradesFileName);
205             FileWriter gfw = new FileWriter(gradesOutputFile.getAbsoluteFile( ), true);
206             BufferedWriter gbw = new BufferedWriter(gfw);
207 
208             //alert user if file doesn't exist
209             if (!outputFile.exists( )){
210                 System.out.println("This class does not exist.");
211 
212                 bw.write("IdNumber,First Name,Last Name, Age");
213                 bw.newLine();
214             }
215             //if the file exists
216             else {
217                 bw.write(s.lastName + ',' + s.firstName + ',' + s.age + ',' + s.idNumber);
218                 bw.newLine();
219                 bw.close( );
220                 int amountOfGrades = amountOfAssignments(className);
221                 String zeros = new String();
222                 for (int i = 0; i < amountOfGrades; i++){
223                     zeros = zeros + ",0";
224                     System.out.println(zeros);
225                 }
226 
227                 gbw.write(s.lastName + ',' + s.firstName + ',' + s.age + ',' + s.idNumber + zeros);
228                 gbw.newLine();
229                 gbw.close( );
230                 addToMasterStudentList(s);
231             }
232         } catch(IOException ex) {
233             System.out.println("Error opening file.");
234         }
235 
236     }
```

Logic

File input

readMasterStudentList( ): Read master student list and return a list of strings to display.

```
258     public static List<String> readMasterStudentList(String username) {
259         String fileName = (System.getProperty("user.dir") + "/data/" + username + "/master_student_list.csv");
260         List<String> studentIDList = new ArrayList<String>( );
261         String studentID;
262 
263         List<String> studentLine = new ArrayList<String>( );
264 
265         try {
266             InputStream ips = new FileInputStream(fileName);
267             InputStreamReader ipsr = new InputStreamReader(ips);
268             BufferedReader br = new BufferedReader(ipsr);
269             String line;
270             int counter = 0;
271 
272             while((line = br.readLine( )) != null) {
273                 String[ ] s = line.split(",");
274                 System.out.println(s[counter]);
275                 studentIDList.add(s[counter]);
276                 studentLine.add(line);
277             }
278 
279 
280             br.close( );
281         } catch(IOException e) {
282             System.out.println(e);
283         }
284 
285         System.out.println("User ID List");
286         for (int i = 0; i < studentIDList.size( ); i++)
287             System.out.println(studentIDList.get(i) + "\n");
288 
289         return studentIDList;
290     }
291 
```

GUI

GUI

Introduction

Designed with the user in mind, to be as intuitive as possible.

Started out as a simple terminal application.

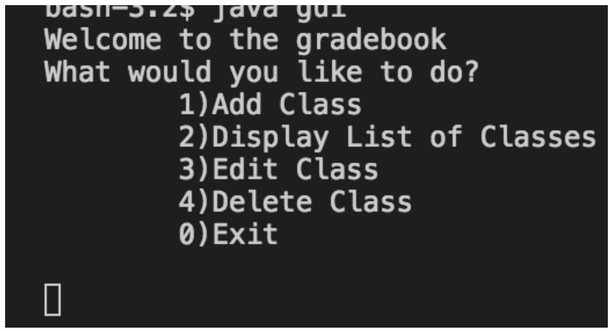
- It only allowed the user to add classes, add students, and edit some basic information.

Slowly implemented a primitive GUI around the terminal application. Though ugly, it was just enough to start working on applets, frames, and buttons to work on the UI.

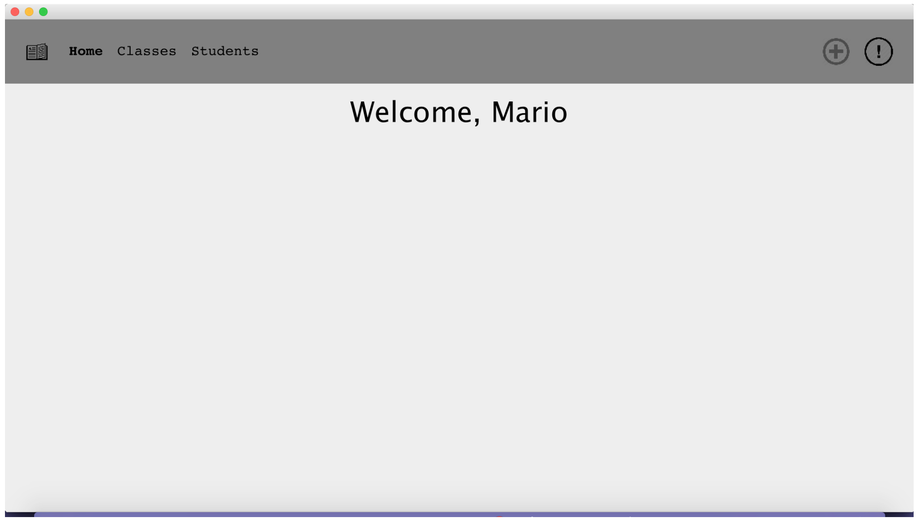
Added finishing touches like custom imported font and logos, located in a resources directory.

GUI

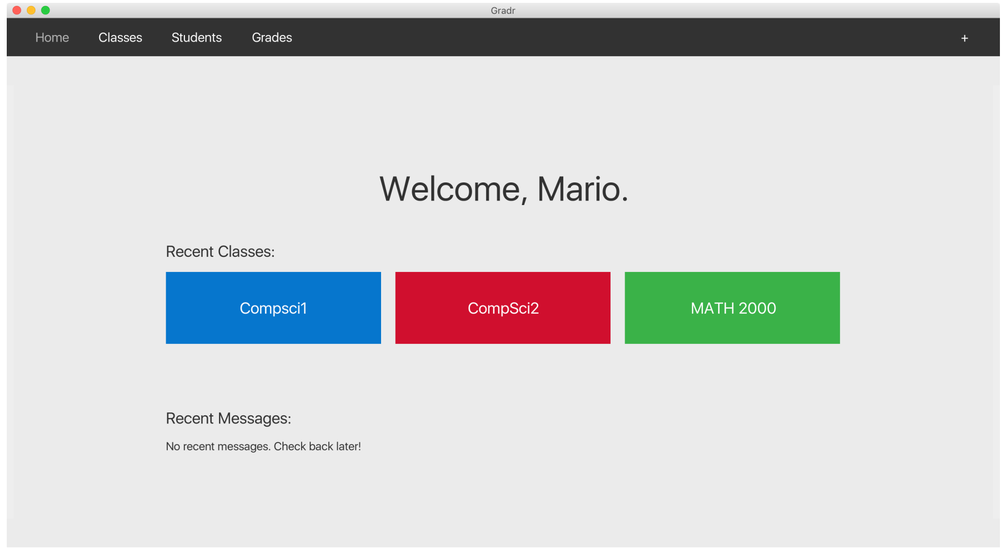
Evolution of the Dashboard



October 10



October 29

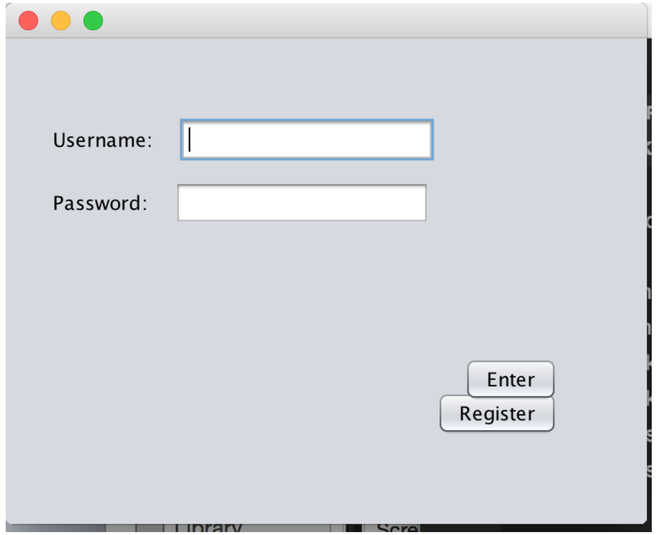


GRADR

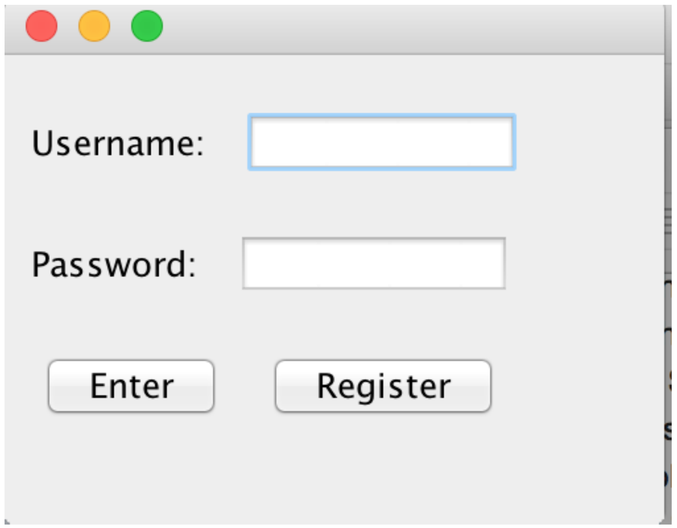
Today

GUI

Evolution of the login screen



October 10



October 29

Today

GUI

The Dashboard

The dashboard is where the GUI lives. You'll find all user functionality here.

It's not too complex; the main Dashboard is a simple Border Layout:

North contains the navigation bar.

South contains the Gradr logo.

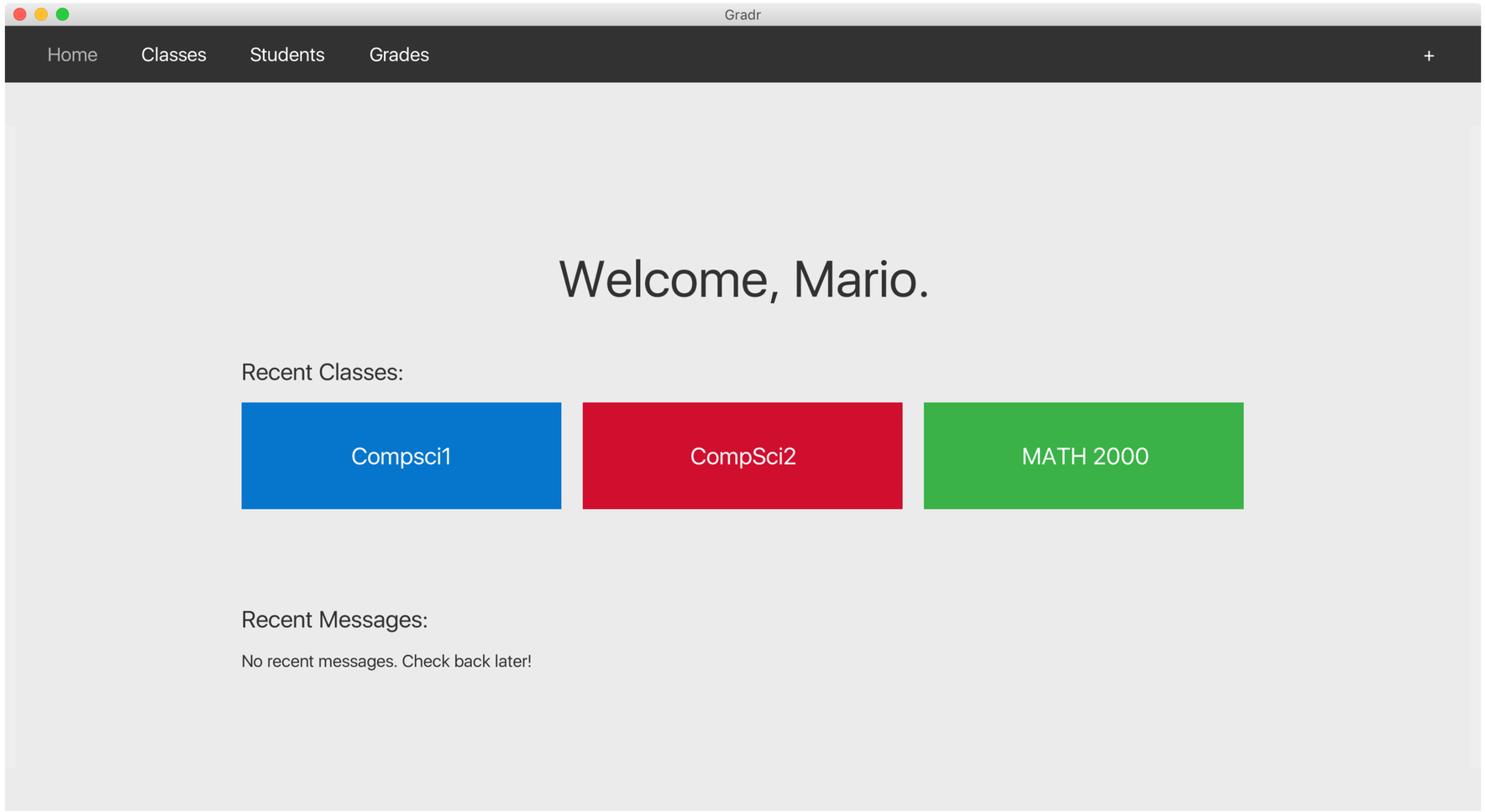
Center contains the dynamic user content.

The swapCenter(JPanel) function receives a panel to display to the user and removes the current for fast switching between pages.

GUI

Home page

The homepage welcomes the user with their first name, then loads a max of 3 recent classes to the Center panel. The labels are large, clickable buttons that link to the referenced Classes page.



GRADR

GUI

swapCenter( )

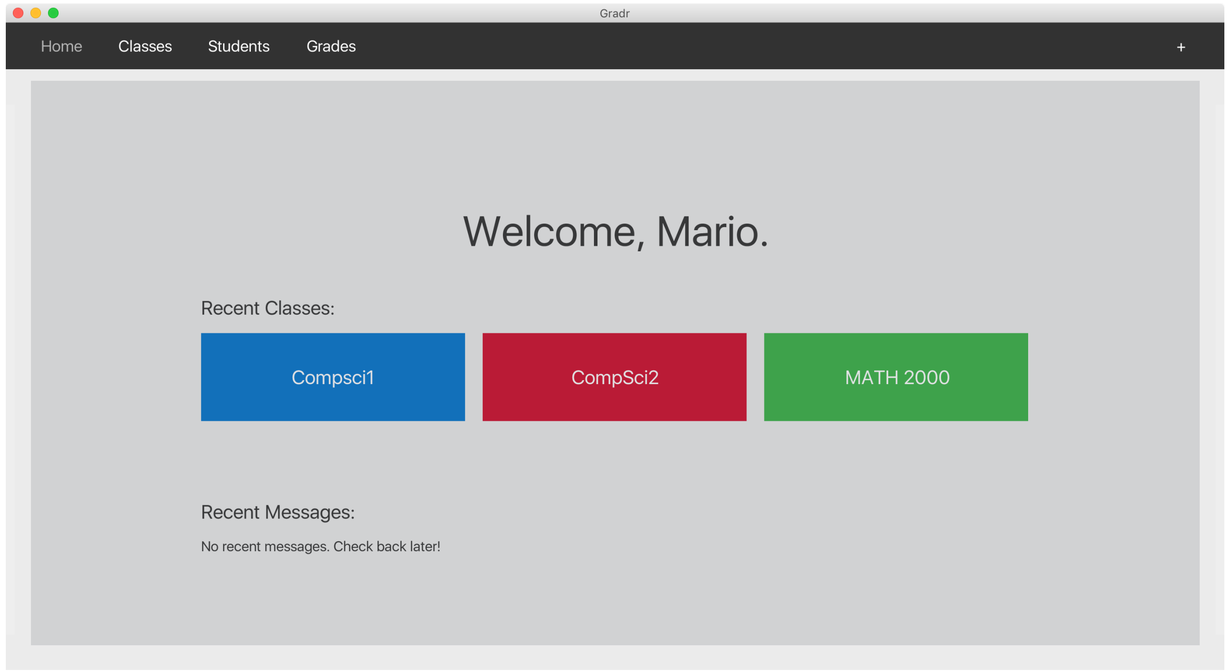
This code swaps the center panel of the Dashboard with any other JPanel, making our code **modular**, and allowing us to easily test each new JPanels as development continued.

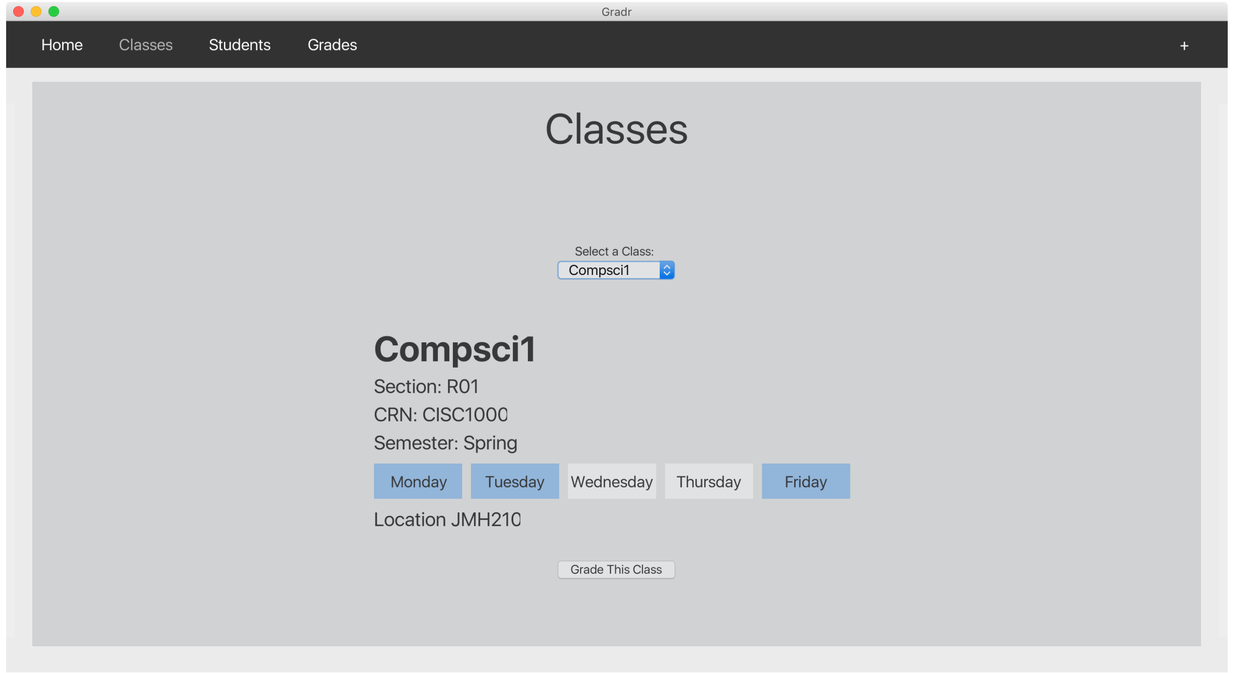
```
433     // receives panel and swaps with the current CENTER screen
434     public void swapCenter(JPanel replacementPanel){
435         BorderLayout layout = (BorderLayout) contentPane.getLayout();
436         if (layout.getLayoutComponent(BorderLayout.CENTER) != null) {
437             contentPane.remove(layout.getLayoutComponent(BorderLayout.CENTER));
438         }
439 
440         contentPane.add(replacementPanel, BorderLayout.CENTER);
441         contentPane.validate();
442         contentPane.repaint();
443     }
444 
```

GUI

swapCenter( )

The shaded area represents the JPanel being swapped when clicking **Compsci1**.





GUI

Action listeners

Each JLabel on the navigation bar has a listener which sets the clicked JLabel's font as highlighted **and** swaps the Center panel for the clicked JPanel.

```
classes.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        home.setForeground(lightTextColor);
        students.setForeground(lightTextColor);
        classes.setForeground(lightSelectedText);
        addButton.setForeground(lightTextColor);
        grades.setForeground(lightTextColor);
        swapCenter(new Classes());
    }
});
```

GUI

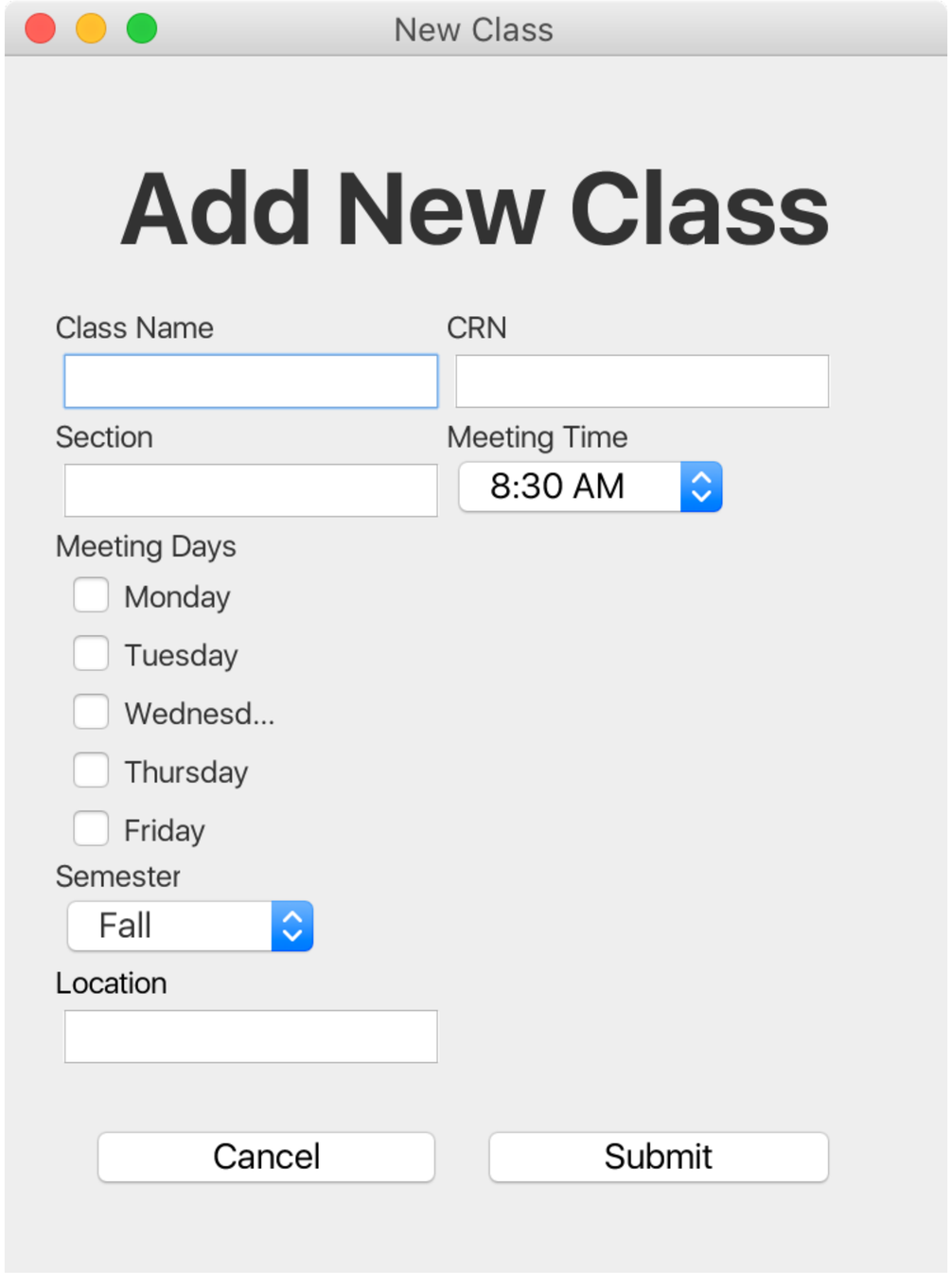
Add page

Add Student

Displays JTextFields that take in student information. This information is eventually compiled into a student object, then saved to the correct files via File IO commands.

Add Class

Displays and performs same operations for class information.



GUI

Submitting information

```
submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        FileIO addStudent = new FileIO();
        String className = (String)classDropDown.getSelectedItem();
        String firstName = fnameField.getText();
        String lastName = lnameField.getText();
        int age =  Integer.parseInt(ageField.getText());
        String id = idField.getText();
        System.out.println(0);
        student newStudent = new student();
        newStudent.setStudentInfo(firstName, lastName, age, id);
        System.out.println(1);
        addStudent.appendStudent(newStudent, className);
        System.out.println(2);
        dispose();
    }
});
```

```
submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        FileIO addClass = new FileIO();
        String nameString = name.getText();
        String crnString = crn.getText();
        String sectionString = section.getText();
        String meetingTimeString = (String)meetingTime.getSelectedItem();
        String semesterString = (String)semester.getSelectedItem();
        String locationString = location.getText();

        // create list of meeting days to pass to classroom
        List<String> meetingDaysList = new ArrayList<String>();
        if(monday.isSelected()){
            meetingDaysList.add("Monday");
        }
        if(tuesday.isSelected()){
            meetingDaysList.add("Tuesday");
        }
        if(wednesday.isSelected()){
            meetingDaysList.add("Wednesday");
        }
        if(thursday.isSelected()){
            meetingDaysList.add("Thursday");
        }
        if(friday.isSelected()){
            meetingDaysList.add("Friday");
        }

        classroom newClassRoom = new classroom(nameString, crnString, semesterString,
            sectionString, meetingTimeString, meetingDaysList, locationString);

        newClassRoom.addClass();
        addClass.createClassInfoFile(newClassRoom);
        dispose();
    }
});
```

GUI

Classes page

The class page displays:

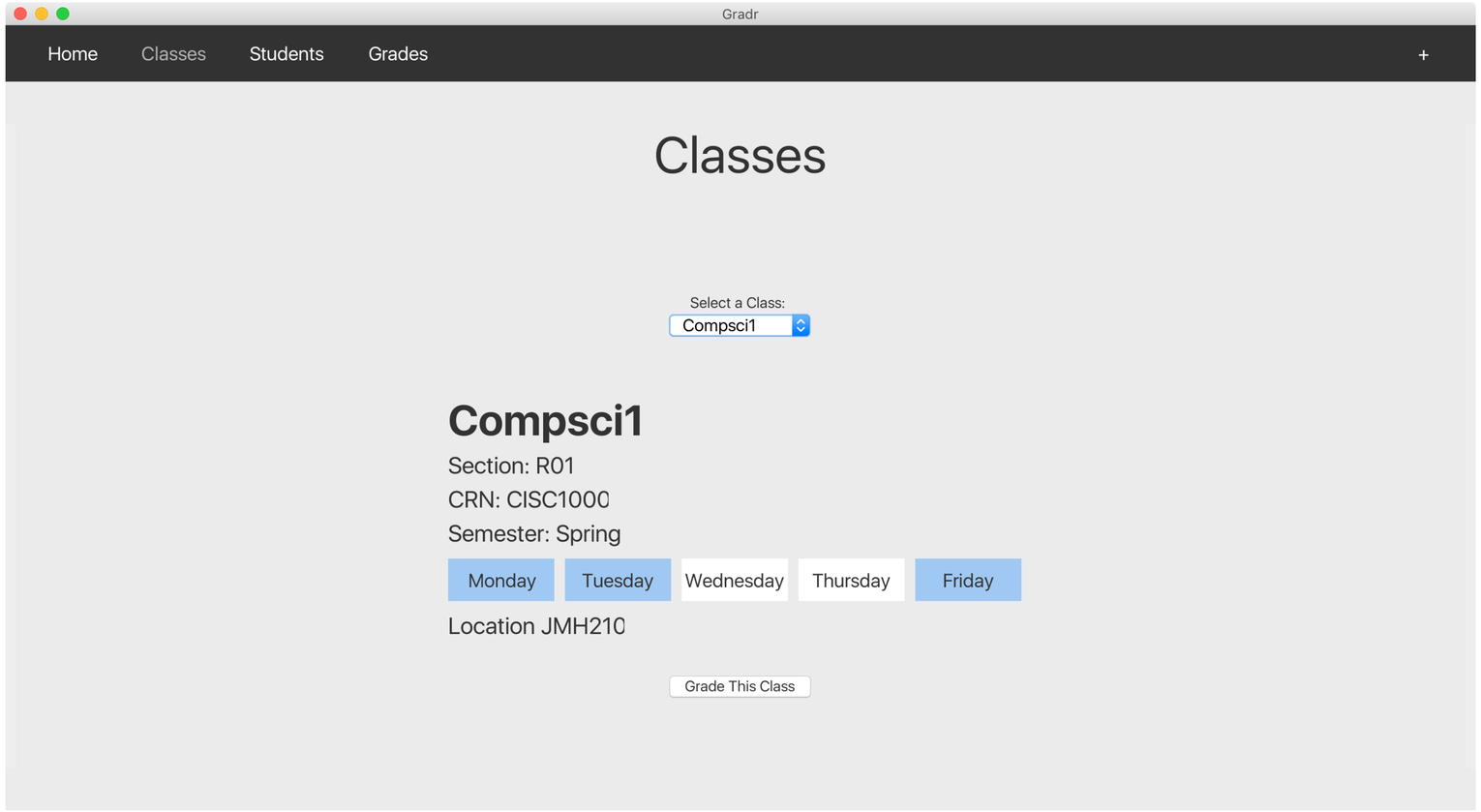
- meeting times
- CRN
- room number

The page consists of JLabels that are updated when the drop down is changed.

- The update function is called when the drop down is changed and uses the File IO function `getClassInfo(className)` to get the class information required.

GUI

Classes page and update( )



GRADR

```
247     private void update(String newClassName){
248         FileIO io = new FileIO();
249         className.setText(newClassName);
250         classroom c = new classroom();
251         c = io.getClassInfo(newClassName);
252         className.setText(c.className);
253         section.setText("Section: " +c.section);
254         semester.setText("Semester: " + c.semester);
255         meetingTime.setText("Meeting Time: " + c.meetingTime);
256         location.setText("Location " + c.location);
257         crn.setText("CRN: " + c.crn);
258         String daysString = new String();
259         for (int i = 0; i < c.daysOfTheWeek.size(); i++){
260             daysString = daysString + c.daysOfTheWeek.get(i);
261         }
262         if(daysString.contains("Monday")){
263             monday.setBackground(daysOfTheWeekSelected);
264         }
265         else{
266             monday.setBackground(daysOfTheWeekDefault);
267         }
268         if(daysString.contains("Tuesday")){
269             tuesday.setBackground(daysOfTheWeekSelected);
270         }
271         else{
272             tuesday.setBackground(daysOfTheWeekDefault);
273         }
274         if(daysString.contains("Wednesday")){
275             wednesday.setBackground(daysOfTheWeekSelected);
276         }
277         else{
278             wednesday.setBackground(daysOfTheWeekDefault);
279         }
280         if(daysString.contains("Thursday")){
281             thursday.setBackground(daysOfTheWeekSelected);
282         }
283         else{
284             thursday.setBackground(daysOfTheWeekDefault);
285         }
286         if(daysString.contains("Friday")){
287             friday.setBackground(daysOfTheWeekSelected);
288         }
289         else{
290             friday.setBackground(daysOfTheWeekDefault);
291         }
292     }
```

GUI

Data table

DataFileTable.java

This class is primarily used to create JTables from CSV files.

Leveraged on the Students page and the Grades Page.

The DataFileTable class is used for determining which CSV file to load.

Key function in this class is **initVectors( )**:

- Loads the appropriate CSV and turns it into a vector called data.
- Vector data is easily converted into a JTable.

GUI

initVectors( )

Loads the appropriate CSV and turns it into a vector called data.

Vector data is easily converted into a JTable to display on Students and Grades pages.

```
23      public void initVectors() {
24
25          String aLine ;
26          data = new Vector();
27          columnNames = new Vector();
28
29          try {
30              FileInputStream fin =  new FileInputStream(datafile);
31              BufferedReader br = new BufferedReader(new InputStreamReader(fin));
32              int location = 0;
33              if (type == 'g'){
34                  String[] parts = datafile.split("/");
35                  for(int i = 0; i < parts.length; i++){
36                    if(parts[i].contains("data")){
37                      location = i + 2;
38                    }
39                  }
40                  List<String> assignmentList = io.assignmentNames(parts[location]);
41                  //int numOfAssignmnets = io.amountOfAssignments(parts[7]);
42                  columnNames.addElement("Last Name");
43                  columnNames.addElement("First Name");
44                  columnNames.addElement("Age");
45                  columnNames.addElement("Id Number");
46                  for(int i=0; i < assignmentList.size(); i++){
47                      columnNames.addElement(assignmentList.get(i));
48                  }
49              }
50              if (type == 'm'){
51                  columnNames.addElement("Id Number");
52                  columnNames.addElement("Last Name");
53                  columnNames.addElement("First Name");
54                  columnNames.addElement("Age");
55              }
56              if (type == 's'){
57                  columnNames.addElement("Last Name");
58                  columnNames.addElement("First Name");
59                  columnNames.addElement("Age");
60                  columnNames.addElement("Id Number");
61              }
62              // extract data
63              while ((aLine = br.readLine()) != null) {
64                  StringTokenizer st2 =
65                  new StringTokenizer(aLine, ",");
66                  while(st2.hasMoreTokens())
67                      data.addElement(st2.nextToken());
68              }
69              br.close():
```

GUI

Students page

Contains three items:

- JLabels.
- Drop-down containing the class list.
- JTable to display student CSV.

Page has a Border Layout.

Utilizes DataFileTable.java to create a JTable loads the Master Student list file.

There is a listener on the drop down to change to the appropriate class list.

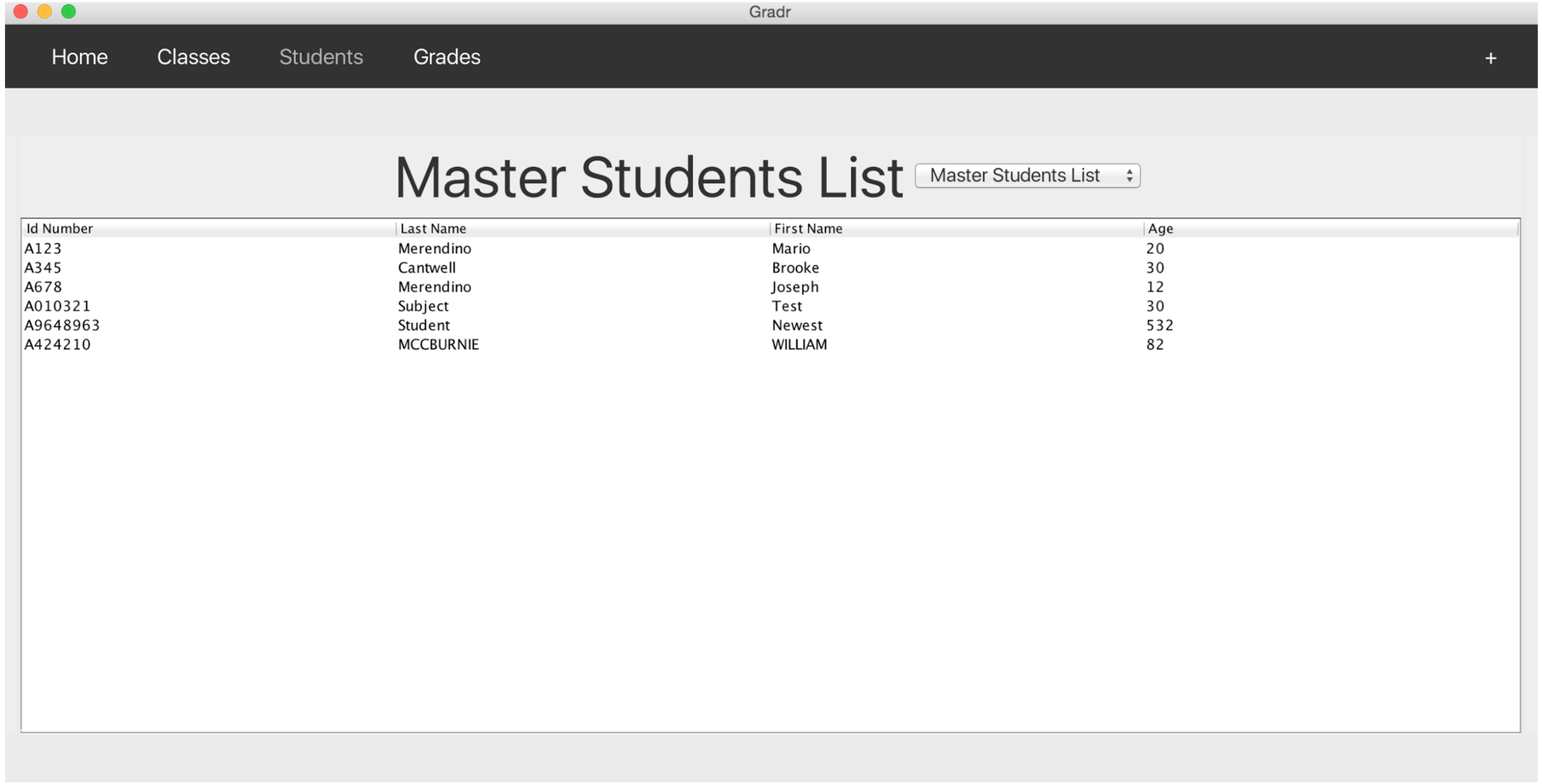
The listener calls the update(classname) which swaps the center panel for the requested class table.

Each student has a line in the students.csv

- FirstName,LastName,FDIN,Age

GUI

Students page



GUI

Grades page

Takes a similar approach to the Students page.

Has a Border Layout.

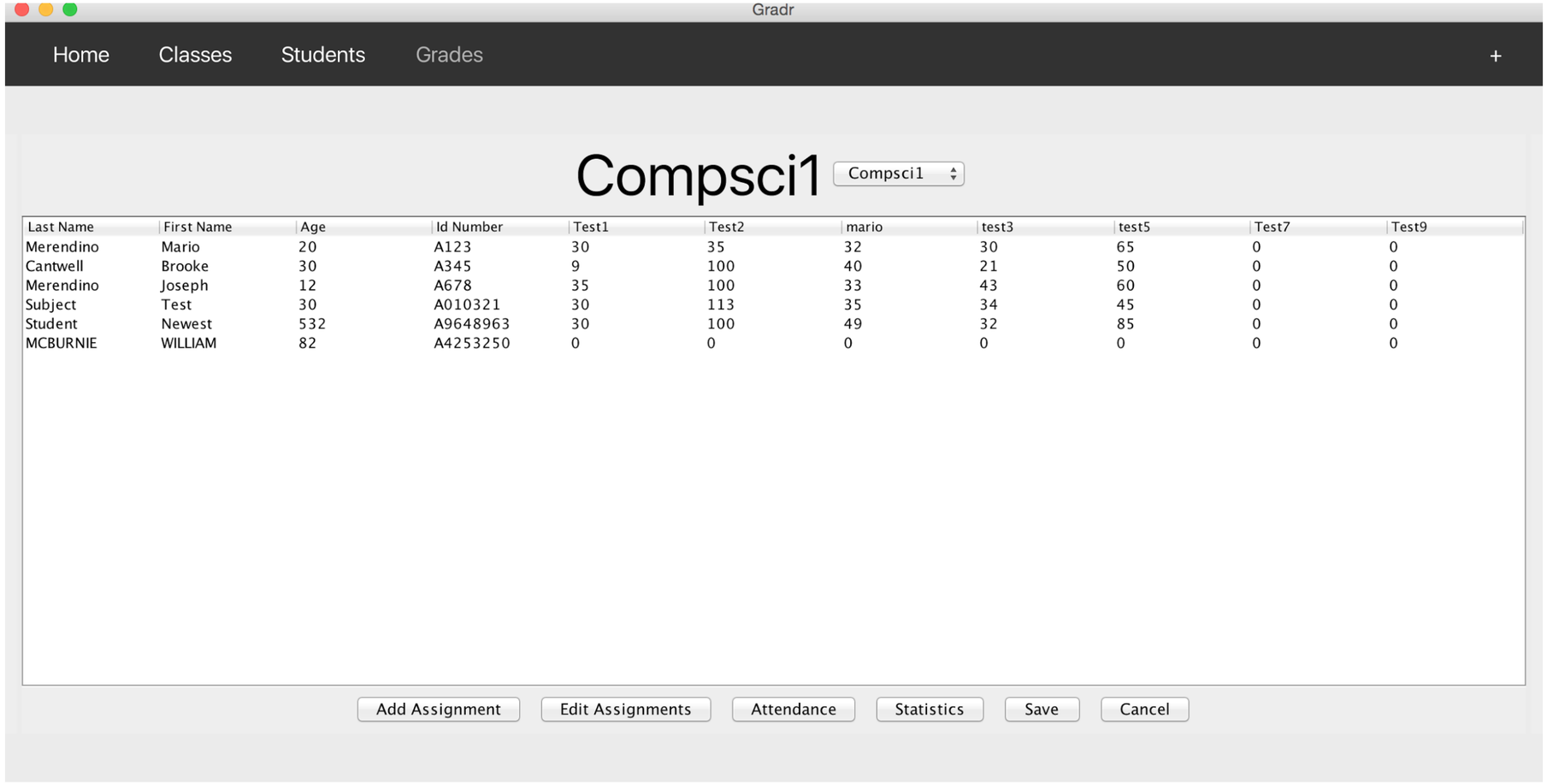
Southern panel contains a row of function buttons.

Listener on the drop-down will load the requested class data from the CSV file.

- The listener calls the **update(classname)** function swaps the center panel with a new JTable.
- The table allows the user to edit grades and save the changes with the save button.
- Each student has a line in the grades.csv file.
- FirstName, LastName, FDIN,Age,0,0,0

GUI

Grades page



GUI

The Grades page save button

The save button converts the JTable to a vector, and performs the FileIO function saveAssignments(vector, className) to save the information to the appropriate CSV.

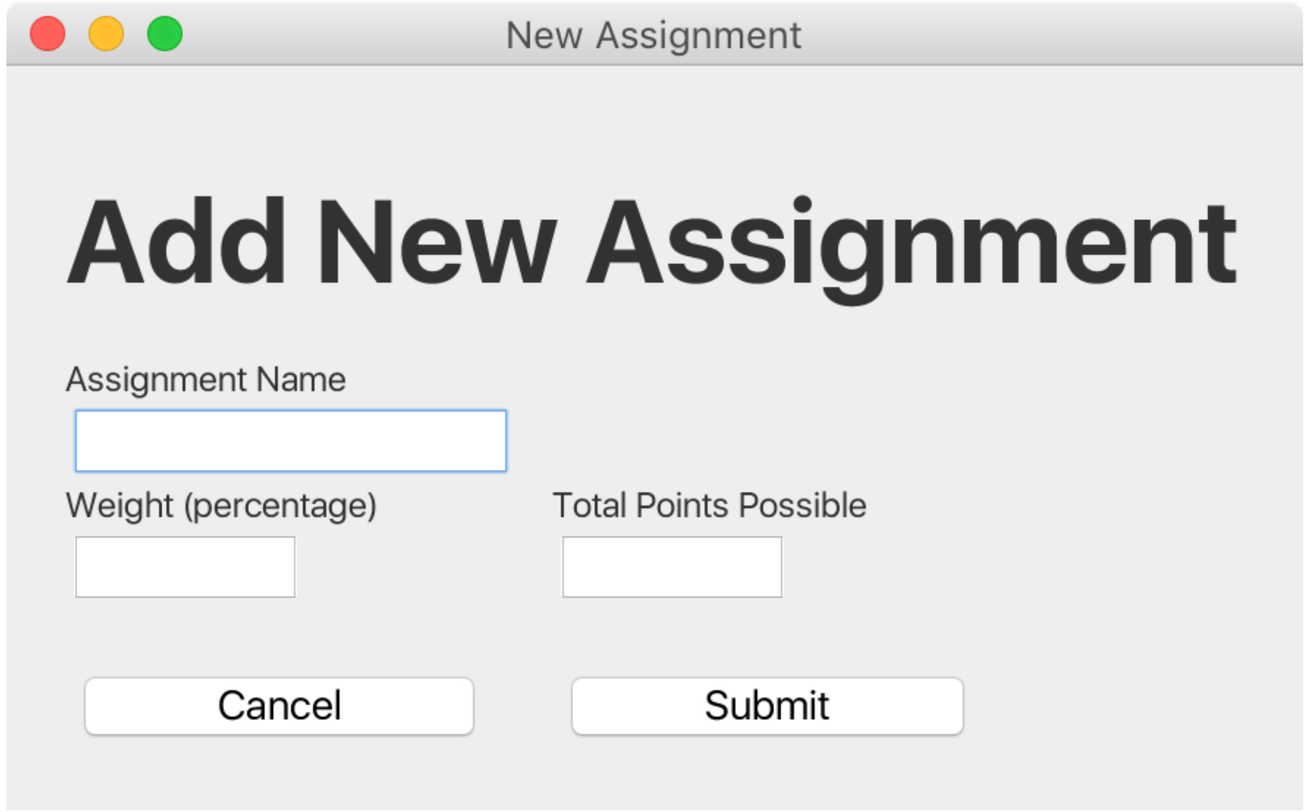
```
saveButton.addActionListener(new ActionListener( ) {
    public void actionPerformed(ActionEvent e) {
        Vector v = new Vector(currentClass.model.data);
        io.saveAssignments(v,(String)classDropDown.getSelectedItem());
        update((String)classDropDown.getSelectedItem());
    }
});
```

GUI

Minor panels and pages

Additional user functionality:

- Stats.
- Attendance.
- Edit assignment.
- Add assignment.



Challenges

Challenges

Communication

More frequent meetings in-person to stay up to date.

Version control

Utilize tools like GitHub to manage builds and versions.

Email was inefficient and confusing.

Learning new styles

Different coding styles made for difficult implementation and stitching of code.

Learning a new language

Learning a new language and using GUI for the first time delayed our launch.

Questions?

Created and edited by Mario Merendino, Brooke Cantwell, and Carlo Mazzoni.