Daniel Santillan

Mrs. Terri Kelly

DE Discrete Structures and Object-Oriented Programming

3 June 2025

222 Final Project Initial Project Description

For my 222 Final Project, I intend to make a functional education portal. This portal can be the backbone for most educational programs, holding users for administrators, teachers, and students and maintaining assignments and grades for various classes. Data is saved between sessions through comma-separated value (CSV) files, which can hold many users, grades, and school systems. The program has many functionalities for schools. Administrators can build customizable schedules and classes, assign students to classes, manage teachers, and create transcripts. Teachers, meanwhile, can grade students, edit their class homepages, and handle disciplinary actions. Lastly, students can submit work for their assignments and view grades. They can also see report cards and simulate their grades. This project is comparable to a tool such as Blackbaud or Schoology.

I started working on this project during the first semester and managed to build a file management system that held user accounts. Additionally, I built a working register and login system that allowed for proper school systems. However, the program was inefficient in determining the user logged in and the school they are a part of. Additionally, many of the functionalities I intended to add to my semester project were not implemented. These will be implemented in the final project, alongside new features such as CSV file management, transcript creation, and disciplinary handling. As I work on this final project, I will focus on file

management and other aspects of the program requiring data storage before implementing the little details that make this program special.

Pseudo Code

Files Package

```
class Constants
        public static final int ADMINISTRATOR_VALUE = 190
        public static final int TEACHER_VALUE = 191
        public static final int STUDENT_VALUE = 192

        public static final int GRADE_WEIGHTS = 290
        public static final int GRADE_PERCENTS = 291
        public static final int GRADE_POINTS = 292

        public static final char DELIMITER_CLASS_CLASS = '大'
        public static final char DELIMITER_CLASS_TEACHER = '张'
        public static final char DELIMITER_CLASS_STUDENT = '伟'
        public static final char DELIMITER_CLASS_GRADING_METHOD = '法'
        public static final char DELIMITER_CLASS_ASSIGNMENTS = '功'
        public static final char DELIMITER_CLASS_GRADES = '等'
        public static final char DELIMITER_CLASS_HOMEPAGE_INFO = '家'
        public static final char DELIMITER_CLASS_END = '习'
end



class DataManagement
        private static User loggedInUser
        private static int currentSchoolID

        private static ArrayList<User> users
        private static ArrayList<School> schools
        private static ArrayList<SchoolClass classes

        public static void setup()
                instantiate users as empty ArrayList<User>
                instantiate schools as empty ArrayList<School>
                instantiate classes as empty ArrayList<SchoolClass>

                instantiate contents as new ArrayList<String>
                set contents to FileWorker.readSchoolFile()

                instantiate schoolNames as new ArrayList<String>
                instantiate schoolIDs as new ArrayList<Integer>

                for (int i = 1; i < size of contents; i++)
                        check contents for the school name by procuring the substring
```

```
                of contents at index i from 0 to an instance of ‘,’
                add the school name to schoolNames
                check contents for the school ID by procuring the substring of
                contents at index i from the instance of ‘,’ onwards, then parse it into
                an integer.
                add the ID to schoolIDs
        end
end

public static boolean addUser(User u)
        add u to users
        return whether the method ran without exceptions thrown
end

private static boolean addNewSchoolAdministrator(Administrator a)
        add a to users
        set loggedInUser to a
        for every School object s in schools:
                if (the s’s ID equals the current school ID) then
                        add Administrator a to s
                end
        end
        return whether the method ran without exceptions thrown
end

public static boolean addNewAdministrator(String username, String firstName, String
lastName, String password)
        instantiate a as new Administrator with username, firstName, lastName, and
        password
        add a to users
        for every School object s in schools:
                if (s’s ID equals the current school ID) then
                        add Administrator a to s
                end
        end
        return whether the method ran without exceptions thrown
end

public static boolean addSchoolAndAdministrator(String schoolName, String
adminUsername, String adminFirstName, String adminLastName, String
adminPassword)
        instantiate schoolID as a random 6-digit integer
        if (addSchool(with new School object using schoolName and schooLID) and
        addNewSchoolAdministrator(with adminUsername, adminFirstName,
        adminLastName, and adminPassword)) then
                return true
```

```
        end
return whether the method ran without exceptions and worked
end

public static boolean addNewTeacher(String username, String firstName, String
lastName, String password)
        instantiate t as new Teacher object with given values
        add t to users
        for every School object s in schools:
                if (s's ID equals the current school ID) then
                        add t to s
                end
        end
        return whether the method ran without exceptions thrown
end

public static boolean addNewStudent(String username, String firstName, String
lastName, String password)
        instantiate s as new Student object with given values
        add t to users
        for every School object sc in schools:
                if (sc's ID equals the current school ID) then
                        add s to sc
                end
        end
        return whether the method ran without exceptions thrown
end

public static boolean addSchool(School s)
        add s to schools
        call FileWorker.writeSchool(s) to write it into the file
        return whether the method ran without exceptions thrown
end

public static boolean addAdministratorToList(Administrator a)
        add a to users
        return whether the method ran without exceptions thrown
end

public static boolean addTeacherToList(Teacher t)
        add t to users
        return whether the method ran without exceptions thrown
end


public static boolean addStudentToList(Student s)
```

```
            add s to users
            return whether the method ran without exceptions thrown
        end

        public static boolean deleteUser(int id)
            for (int i = 0; i < size of users list; i++)
                if (user at index i's ID equals id) then
                    for every School s in schools
                        if (s's ID equals user's student ID) then
                            s.deleteUser(user at index)
                        end
                    end
                end
            end
            return whether the method ran properly and without exceptions
        end

        public static boolean login(String username, String password)
            for every User u in users:
                if (u's username equals username and u's password equals password) then
                    setLoggedInUser(u)
                    return true
                end
            end
            return false
        end

        setters and getters for loggedInUser

        public static void logOutUser()
            nullify loggedInUser and make currentSchoolID negative
        end
end



class FileWorker
        private static File schoolList

        public static void makeFile()
            instantiate schoolList as a new File with filename SchoolList.csv

            if (schoolList does not exist) then
                instantiate bWriter as a new BufferedWriter for schoolList
                use bWriter to write in the csv values of School Name,School ID
                close bWriter
```

```
        end
    end

    public static boolean writeSchool(School s)
        instantiate contents as a new ArrayList<String> from readFile(schoolList)
        instantiate bWriter as a new BufferedWriter for schoolList
        for every line in contents:
            use bWriter to write in every line
        end
        use bWriter to write in the new school name and ID
        close bWRiter
        return whether the method worked without exceptions thrown
    end

    public static ArrayList<String> readFile(File f)
        instantiate contents as a new and empty ArrayList<String>
        instantiate reader as a new BufferedReader for f
        while the next line is not empty
            use reader to read the line and add its characters to contents
        end
        close the reader
        return contents
    end

    public static ArrayList<String> readSchoolFile()
        return readFile(schoolFile)
    end

    public static boolean removeLine(File f, int id)
        instantiate contents as return-value of readFile(f)
        instantiate forgetIndex to -1
        for (int i = 0; i < size of contents; i++)
            if the contents value at index i has the id then
                set forgetIndex to i
            end
        end
        instantiate bWriter as new BufferedWriter for f
        for (int i = 0; i < size of contents; i++)
            if i does not equal the forgetIndex then
                write in the line at index i
            end
        end
        return whether the method ran without exceptions thrown
    end
end
```

Graphics Package

```
class Frame extends JFrame
        private CardLayout cl
        private Container container

        public Frame()
                set title of the frame
                set size to the screen size
                set default close operation to JFrame.EXIT_ON_CLOSE
                set focusable

                instantiate cl as new CardLayout
                instantiate container as getContentPane()
                set container layout to cl
                prepareCardLayout()
        end

        private void prepareCardLayout()
                instantiate new JPanels for every panel of the program
                instantiate new JButtons for every JButton that swaps between pages
                call modifyButtons() for each button
                call addChangePageButtons() for each JPanel to add buttons that swap between
                pages
                instantiate al as new ActionListener
                        public void actionPerformed(ActionEvent e)
                                establish how to change pages for every JButton
                                for the button that registers accounts or logs in users, procure
                                information from the text fields of the JPanels, complete
                                some input handling, then send to the DataManagement class
                        assign JButtons with ActionListener al
                        add JPanels to container
        end
end


class GraphicsConstants
        public static final Color COLOR_BG_HEADER = new Color(3, 112, 180)
        public static final Color COLOR_BG_MAIN = new Color(216, 224, 232)

        public static final Font FONT_BUTTON = new Font("Roboto", Font.PLAIN, 30)
        public static final Font FONT_ROBOTO_B30 = new Font("Robtoto", Font.BOLD, 30)
        public static final Font FONT_ROBOTO_B50 = new Font("Roboto", Font.BOLD, 50)
```

```
        public static final Dimension DIMENSION_TEXTFIELD_DEFAULT = new
        Dimension(800, 75)
end


class GraphicsHelpers
        public static boolean isPasswordValid(String password)
                if password has whitespace then
                        return false
                end
                else if password has length less than 10
                        return false
                end
                else if password has length greater than 50
                        return false
                end
                else
                        instantiate hasDigit as a boolean set to false
                        instantiate hasLowercase as a boolean set to false
                        instantiate hasUppercase as a boolean set to false
                        instantiate hasSpace as a boolean set to false
                        for (int i = 0; i < length of the password; i++):
                                if (the password's character at index i is uppercase) then
                                        set hasUppercase to true
                                end
                                if  (the password's character at index i is lowercase) then
                                        set hasLowercase to true
                                end
                                if (the password's character at index i is a digit) then
                                        set hasDigit to true
                                end
                                if (the password's character at index i has a space) then
                                        set hasSpace to true
                                end
                        end
                        if (hasUppercase is false or hasLowercase is false or hasDigit is false or
                        hasSpace is true)
                                return false
                        end
                end
                return true
        end

        public static void modifyButton(JButton button, int width, int height)
                set button's background to COLOR_BG_HEADER
                set button's foreground to white
```

```
            set button's font to FONT_BUTTON
            set button's size to a new Dimension with width width and height height
        end
end


JPanel extensions (template)
        Constructor
            set layout to a new BorderLayout
            prepareNorthPanel()
            prepareCenterPanel()
        end

        private void prepareNorthPanel()
            instantiate northPanel as new JPanel
            set northPanel height to 75 pixels
            instantiate header as new JLabel with an image corresponding to the page header
            add header to northPanel
            add northPanel at BorderLayout.NORTH
        end

        private void prepareCenterPanel()
            instantiate sl as a new SpringLayout
            instantiate centerPanel as a new JPanel with layout sl
            add various components
            put constraints on the components through sl
            add centerPanel at BorderLayout.CENTER
        end

        public void addChangePageButtons(JButton button)
            (The number of buttons will vary by page)
            (if needed) instantiate southPanel as a new JPanel
            add the buttons to southPanel
            add southPanel at BorderLayout.SOUTH
        end
end
```

Overview of Pages:

- Homepage
    - Title panel
    - Terms and Conditions
    - Register
    - Login
- Admin
    - Homepage
    - Manage Schedule

    - Add Users
    - Add Classes
    - Delete Users
    - Delete Classes
    - Manage Teachers
    - Manage Students
    - Manage Classes
    - Edit Profile
- Teacher
    - Homepage
    - Class Homepage – will do a JComboBox to select class homepage user wants to view
    - Edit Class Homepage
    - Grade Assignments – will likely use JTables
    - Manage Assignments
    - Manage Students
    - View Rosters – Again, will use JComboBox to select which class user wants to view
    - Edit Profile
- Student
    - Homepage
    - Class Homepage
    - View Grades
    - View Rosters
    - Grade Simulator – similar to my earlier project from first semester
    - Edit Profile
- Each page will have specific components and may need additional helper methods. They may also make necessary the addition of other classes (e.g., the grading pages will require a Calculations class)

Objects Package

class User

    private String username
    private String firstName
    private String lastName
    private String password
    private int id
    private int schoolID

    public User(String username, String firstName, String lastName, String password, int id, int schoolID)
        set object's username to username
        set object's firstName to firstName
        set object's lastName to lastName
        set object's password to password
        set object's id to id

```
            set object's school ID to id
        end

        public User(String username, String firstName, String lastName, String password, int
        schoolID)
            call above constructor but make id a random 6-digit number
        end

        public boolean isAdministrator()
            return false
        end

        public boolean isTeacher()
            return false
        end

        public boolean isStudent()
            return false
        end

        getters for all fields, setters for username, firstName, lastName, and password
end

class Administrator extends User
        have the same constructors as User, just call super and this when needed

        Override public boolean isAdministrator()
            return true:
        end
end

class Teacher extends User
        have the same constructors as User, just call super and this when needed

        Override public boolean isTeacher()
            return true:
        end
end

class Student extends User
        have the same constructors as User, just call super and this when needed

        Override public boolean isStudent()
            return true:
        end
end
```

```
class School
        private String name
        private int schoolID
        private File userList
        private File classList
        private ArrayList<Administrator> admins
        private ArrayList<Teacher> teachers
        private ArrayList<Student> students
        private ArrayList<SchoolClass> classes

        public School(String name, int schoolID)
                set object name to name
                set object schoolID to schoolID

                instantiate admins, teachers, students, and classes as empty lists
                instantiate userList as a new File with name SchoolUsers_[ID].csv
                instantiate classList as a new File with name SchoolClasses_[ID].csv

                if (userList does not exist) then
                        create a new file with userList's filename
                        instantiate BufferedWriter bWriter for userList
                        write "Username,First Name,Last Name,Role, User ID, Password" for csv
                        close bWriter
                else
                        addExistingUsers
                end

                if (classList does not exist) then
                        create a new file with classList's username
                        setup the file with delimiters through a BufferedWriter
                        close bWriter
                else
                        addExistingClasses
                end
        end

        public boolean AddAdministrator(Administrator a)
                add a to admins
                writeInUser(a, "Administrator")
                return whether the method ran without exceptions thrown
        end

        public boolean AddTeacher(Teacher t)
                add t to teachers
```

```
        writeInUser(t, “Teacher”)
        return whether the method ran without exceptions thrown
end

public boolean AddStudent(Student s)
        add s to Student
        writeInUser(t, “Student”)
        return whether the method ran without exceptions thrown
end

private void writeInUser(User u, String role)
        instantiate contents as the result of FileWorker.readFile(userList)
        instantiate BufferedWriter bWriter for the userList file
        for (int i = 0; i < size of contents; i++)
                write the contents of the file to the file by line
        end
        write in the new user with their role in the format of the csv file
        close bWriter
end


private void addExistingUsers()
        instantiate contents as the result of FileWorker.readFile(userList)
        for (int i = 1; i < size of contents; i++)
                instantiate username, firstName, lastName, role, and id Strings
                based on the values of each line in the csv file

                if (role is administrator)
                        instantiate a new administrator based on procured values and
                        assign it to admins
                        add this administrator to DataManagement’s user database
                end

                else if (role is teacher)
                        instantiate a new teacher based on procured values and assign
                        it to teachers
                        add this teacher to DataManagement’s user database
                end

                else if (role is student)
                        instantiate a new student based on procured values and assign it to
                        students
                        add this student to DataManagement’s student database
                end
        end
end
```

```
        public void deleteUser(User u)
                if (u is administrator)
                        Search through admins and if user ID matches an ID, remove that
                        admin from the list
                end
                else if (u is teacher)
                        Search through teachers and if user ID matches an ID, remove that
                        teacher from the list
                end
                else if (u is student)
                        Search through students and if user ID matches an ID, remove that
                        student from the list
                end
                remove the line of data with the student from the userList file through
                FileWorker.removeLine()
        end

        getters for name, school ID, and list of users



class SchoolClass
        private String name
        private String block
        private int teacherID
        private ArrayList<Integer> studentIDs
        private int gradingMethod
        private ArrayList<Assignment> assignments
        private ArrayList<String> weightCategories

        public SchoolClass(String name, int block, int teacherID, ArrayList<Integer> studentIDs,
        int gradingMethod, ArrayList,Assignment> assignments, ArrayList<String>
        weightCategories
                instantiate fields based on given information
        end

        public void addAssignment(Assignment a)
                add a to assignments
                end

                public void deleteAssignment(String assignmentName)
                        for every Assignment a in assignments
                                if a's name equals the assignmentName then
                                        remove a from the list
                                end
```

```
            end
        end

        public void addStudent(Student s)
            add s's student ID to studentIDs
            for every Assignment a in assignments:
                a.addStudentDisplacement(some index - unsure)
            end
        end

        public void deleteStudent(Student s)
            for every student ID in studentIDs
                if s's student ID equals a student ID then
                    remove the student from the studentIDs list
                    for every Assignment a in assignments
                        a.removeStudentDisplacement(some index – WIP)
                    end
                end
            end
        end


class Assignment
    private String name
    private int points
    private String weightCategory
    private ArrayList<Double> grades

    public Assignments(String name, int points)
        assign name and points to respective values
        instantiate grades with empty list
    end

    public Assignment(String name, String weightCategory)
        assign name and weightCategory to respective values
        insantiate grades with empy list
    end

    public boolean setGrade(int index, double grade)
        set value of grades at index index to grade
        return whether the method worked witout exceptions thrown
    end

    public boolean addStudentDisplacement(int index)
        add a null value at grades for index index
        return whether the method worked without exceptions thrown
```

end

public boolean removeStudentDisplacement(int index)
remove the value of grades at index
return whether the method worked without exceptions thrown
end

setters and getters for name, point, and weightCategory

Run Package

class Run
void main(String[] args)
instantiate frame as new Frame object
set frame to visible
end
end