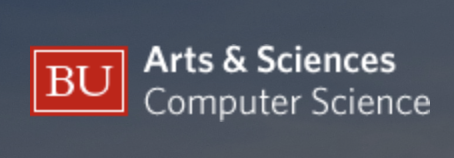
****

**CS 591 P1 FINAL PROJECT**

**GRADING SYSTEM**

**INSTRUCTOR – Christine Papadakis-Kanaris**

|  |  |
| --- | --- |
| **TEAM MEMBERS** | **EMAIL IDs** |
| Alimkhanov, Yernur | yernura@bu.edu |
| Gao, Tian | gaotian@bu.edu |
| Duan, Lian | lduan@bu.edu |
| Das, Yashvardhan | yashvdas@bu.edu |

1. **INTRODUCTION**

* This document explains the motivations, technical functionalities and the overall design process related to the development of our final project related to the course of CS 591 P1 – Object-Oriented Design and Development in Java.
* The project has been developed over a period of weeks with detailed focus provided to different aspects of the, that include but not limited to the – frontend, backed, interface development, documentation, testing, debugging and the final presentation.
* This report is prepared keeping in mind the requirements of the design document associated with the overall software implementation of the project.

1. **REQUIREMENTS ANALYSIS**

* The primary thought process for the software implementation and design process is concerned with understanding the key requirements of our Professor regarding a computerized grading system.
* The motivation behind the successful implementation of our project is based on cent percent alignment of her requirements with our overall class design and each class’ technical functionality.
* As part of the requirement analysis, the process involved inquiring our Professor from time to time about her precise specifications that she wanted to have in her grading system application.
* Some of the primary objectives were to understand the following aspects and subsequently working on them:
  + which features exactly were lacking in the current system,
  + which features present in the current system needed improvisation and,
  + which features present in the current system could be deleted to significantly boost the overall functionality and robustness.

1. **DESIGN OF GRAPHICAL USER INTERFACE CLASSES**

* The following paragraphs explain the functionality of each Graphical User Interface (GUI) class that has been implemented in the application designed.
  + **AddCoursePage.java**
    - This page produces a window that provides the Professor the functionality of adding a new course.
    - Options are provided to choose the name of the course, select the requisite semester in which the course has to be incorporated and, choose from one of the templates based on which the course is to be designed.
    - Buttons are also provided that with options denoting whether to head to the next page, previous page and/or to register specific students into the course’s class.
  + **AddStudentPage.java**
    - This page provisions a GUI-based window for the user to fill up the details of the prospective student who is to be registered for the particular course.
    - The respective details that can be entered by user about the student involve the first, middle and last names, the unique student ID, the unique email associated with the student.
    - One more important option that comes into play includes choosing the academic level/type of the student, i.e., selecting whether the student is an undergraduate or a graduate-level one.
    - The buttons available in this page include the options to confirm or rescind the entered details.
  + **AssignmentAddtionPage.java**
    - This page provisions the user to enter the specifics of the required assignment type that is supposed to be created. The name of the assignment is the first thing the user input.
    - The user can select which type of assignment this is, either from the pre-defined categories, or can create a new category for the particular assignment.
    - An input box is provided to enter the total score related to the specific assignment. A crucial feature in this page is the provision to select the scoring criteria for the respective assignment.
    - The buttons provided at the bottom of the page allow the user to either go back to the earlier page or to finish up adding the respective assignment’s specifics.
  + **CourseStructurePage.java**
    - This page provides the user the choice of selecting the relevant category of the concerned course.
    - The option of providing the requisite percentage for the same is also provided.
    - Buttons present at the bottom of the page provide the users options to either update the markup for the course or to get back to the previous page.
  + **DeleteAssignmentPage.java**
    - This page provisions the user to remove a specific assignment from the concerned course.
    - There is a drop-down list made available for the user to choose the particular assignment that needs to removed.
    - Probable options in the drop-down menu include but not limited to – home-works, midterm, final-exam, project, etc.
    - The buttons provided at the bottom-half of the page allow the user to select the ‘delete’ option button to remove the specific assignment, and the option of navigating to the previous page using the ‘back’ option button.
  + **LoginPage.java**
    - This page is meant for the user to log in into the application.
    - The login application credentials are accessible only to our Professor, who, is meant to be the sole user of this application.
    - The username and password are unique for the Professor.
    - An incorrect combination will result a display of an error window.
  + **MainPage.java**
    - This is the central and most informative page of the application. It provisions a range of buttons and links to navigate to the respective pages.
    - One important feature that this page provides is the overall average, median and standard deviation of the marks that are associated with home-works, project or exam.
    - A check-box is also provisioned to signify whether the particular assignment will be graded or not. After selecting the required criteria, the user is provided an ‘apply’ button to execute the scenario.
    - Towards the right side of the page, links are provided in the form of buttons to navigate to other pages as well as some other related functionalities.
    - The option of curving the grade by a certain integer value is also provided. The user is also provisioned to navigate to another specific page that has the functionality of modifying/updating the academic status of students.
  + **SelectCoursePage.java**
    - This page provisions the user to choose an existing course she is supervising or create a new course that might be undertaken in the near future.
    - The choices of selecting the respective semester and course name are provided with the help of drop-down menus.
    - The button-link for entering the details of marks and other specifics is provided. The option of creating a new course through a similar button-link is also provided.
    - One key feature of this page is the provision of template management option that is also provided through the presence of a button-link.
    - The user also the option to log out from the application from this page.
  + **StudentManagePage.java**
    - This page provisions the user to alter/update the academic status of a student concerned with the required course.
    - The user can add a new student to the course by entering the respective details, remove a particular student from the concerned course or, freeze the specific details associated.
    - The ‘back’ button-link allows the user to navigate to the previous page.
  + **TemplateManagementPage.java**
    - This is one of the most important features designed in the application.
    - This page allows to assign either a pre-determined template to a course or create a new one and then assign it to the respective course.
    - The ready-made templates can be selected through the drop-down menu that is available in the page.
    - There is also an option to delete an existing template.
    - The ‘update’ button-link is used to submit the functionalities of the template.

1. **DESIGN OF ENTITY CLASSES**

* The following paragraphs explain the functionality of each entity class that has been implemented in the application designed.
  + **Assignment.java**
    - This is one of the fundamental blocks programmed in designing the overall application structure.
    - It contains information in the form of class variables and methods that include the name and weightage of the particular assignment, along with the percentage of the category to which the assignment belongs to.
    - A unique ID is associated as a member variable in this class to pinpoint to the specific assignment.
    - A list has been programmed to contain the respective grades.
    - Relevant setters and getter-methods have been implemented for the requisite purpose.
  + **Category.java**
    - This class has been implemented to define each section of the work that is to be graded by the user.
    - The member variables and methods include those required for naming the required category and assigning the specific weightage (percentage) associated with it.
    - Appropriate setters and getter-methods have been programmed for the requisite purpose.
  + **CategoryPercent.java**
    - This class is quite closely associated with respect to the functionality of the above-mentioned class.
    - The functionality of this class also includes the instance of the overall course that is designed in the application.
    - As such, this class can have different values for different courses, according to the user’s requirements.
    - A list of the relevant assignments is included in this as one of the instance variables, alongside others.
    - Appropriate setters and getter-methods have been programmed for the requisite purpose.
  + **Course.java**
    - This class basically defines the core functionality of the respective class that the user has designed according to her requirements.
    - The constructor of this class instantiates the components such as the names, students associated and the specific semester to which the course has been allocated by the user.
    - Relevant setter and getter-methods have been programmed to serve their respective purpose.
  + **Grades.java**
    - This class is designed to cater to the functionality of assigning the relevant grades based on specifics assigned by the user.
    - The instance variables include a numeric (double) type for initialization of the grade and, a Boolean variable to check whether or not the specific component has been graded or not.
    - The constructor of this class instantiates the respective objects of the Student and Assignment classes.
    - Relevant setters and getter-methods have been implemented for the requisite purpose.
  + **GraduateStudent.java**
    - This is a child class of the Student superclass. As such, it inherits properties from the Student class.
    - The constructors of this class include the instance variables of the name, surname, BU ID and email associated with the graduate student.
    - The course associated with the graduate student is also taken into account.
  + **PhdStudent.java**
    - This is a child class of the Student superclass. As such, it inherits properties from the Student class.
    - The constructors of this class include the instance variables of the name, surname, BU ID and email associated with the PhD student.
    - The course associated with the PhD student is also taken into account.
  + **UndergraduateStudent.java**
    - This is a child class of the Student superclass. As such, it inherits properties from the Student class.
    - The constructors of this class include the instance variables of the name, surname, BU ID and email associated with the undergraduate student.
    - The course associated with the undergraduate student is also taken into account.
  + **Professor.java**
    - This class is used to define the informative functionalities of the user (Professor, in this case).
    - The constructor of this class instantiates the name, surname, login ID and password of the user.
    - Appropriate setters and getters are used in this class.
  + **Semester.java**
    - This class defines the functionalities related to relevant semesters.
    - Getters and setters for setting courses in the semester are programmed in the code.
    - A string-based tag and an integer-based ID is associated with this class for identification in the database.
  + **Student.java**
    - This is one the Parent classes programmed in the code. This class provisions itself as a base class for different types of students designed in other class (e.g., undergrad, grad, PhD).
    - The crucial functionalities covering the overall letter grade and the total score associated with a student is programmed in this class design.
    - The constructors of this class instantiate the instance variables that include the name, surname, BU ID, email, course and the list of grades associated with the student.
    - Appropriate setters and getters have been implemented for the required purpose.
  + **Template.java**
    - This class has been designed to assign the overall structure of the specific course.
    - The functionality of this class is to assign the weightage/percentage distribution of the different components of a specific course.
    - Appropriate setters and getters have been implemented for the required purposes.

1. **CHOICE OF DATABASE IMPLEMENTATION**

* We have utilized Java Persistence API (JPA) for the implementation of the database part.
* The reasons for choosing JPA over other alternatives are listed as follows: -
  + Ease of one-time mapping between database tables and domain model.
  + Portability of writing queries.
  + Automated provision of type and parameter handling.
  + Provision of write-behind optimization.
  + Availability of caching.
  + Simple integration with GUI.