CS 319 / Object-Oriented Software Engineering Term Project:

“Mastering Bilkent”

System Design Report

Project Group 1.A:

- Ertan ADAY
- Furkan Arif BOZDAĞ
- Rümeyza DİNÇER

Course Instructor:

- Bora GÜNGÖREN

Teaching Assistant:

- Gülden OLGUN

Table of Contents

1. Introduction

1.1 Purpose of the System

Mastering Bilkent is a desktop application intended to help students of Bilkent University to study and understand the content of their scheduled courses with supplementary materials such as pdf documents, videos and quizzes provided by their instructors of each course. Although, the main target of the application is Bilkent student, with the permission of instructor courses shall be accessed by public. Therefore, our main aim is to become a desired place for self-education with academic level of quality.

1.2 Design Goals

We first examine the design goals of the system before move on to system architecture.

User Friendliness & Ease of Use: Since Mastering Bilkent is an educational application, users shall adapt our application easily. We don't aim to serve a complicated and not understandable application. Thus, our application considers user friendliness as main criteria. We will provide an easy to learn user interface. Our application is planned to have different layers for instructors and students as we discuss in the analysis stage. Users shall easily access the courses they are looking for via the search bar. A student can access a course which s/he registered from the list on his/her homepage and instructor shall access a course which s/he created from her/his homepage. Users shall easily understand how to use course-related activities (attending/creating a quiz, reading/ uploading documentation or watching/uploading video files) by the simple menus and instructive buttons clickable via mouse.

Modifiability: In Mastering Bilkent, students should be able to change their personal information such as profile pictures, addresses, and other general information about themselves. Moreover, Instructors will also change personal information about themselves, and also they will modify their course pages very often. Therefore, Mastering Bilkent must provide a modifiability.

Availability: Mastering Bilkent is a non commercial enterprise application, it must provide round-the-clock services. That is, our application need to work 24 hours, 7 days, with low amount of downtime per year. It should have a maximum downtime of 90 hours per year. This is why our application designed to satisfy high availability criteria.

Reliability: Mastering Bilkent must keep the data, which is uploaded by both students and instructors, without corrupting it. Since this application can be used by an instructor as an official course homepage, any error that appears related to the course documents can't be accepted. If a student attempts an assignment quiz, it's result should be sent to the instructor correctly. Moreover, Mastering Bilkent is an educational application, any corruption in the course documents or course activities would cause the latency in educational progress. Thus, Mastering Bilkent will be implemented in a reliable way.

Good Documentation: Our application will contain substantial amount of documents which are in audio, video, or text formats. In order to provide a good service to users , Mastering Bilkent should be documented in a good manner. That is why the design should include good documentation.

Scalability: Since Mastering Bilkent is an educational application that will separate the contents by its semesters and years, it should handle the growing amount of data and work. In order to be an efficient educational application, Mastering Bilkent should a good scalability property.

1.3 Trade-offs

Scalability vs. Long Term Reliability: Mastering Bilkent will not store all the growing data in its database and manage it in terms of scalability efficiency. In order to provide sufficient scalability, there will be some trade-offs from long term information.

Good Documentation vs. Memory: In order to provide a good documentation for different type of documents in the forms of text, video, or audio, our application will use substantial amount of memory. So there will be memory tradeoffs in order to provide a good documentation.

User Friendliness & Ease of Use vs. Functionality: It is very important for Students and Professors to be effective and not waste their time navigating around the application. Therefore, Mastering Bilkent will be a user friendly and easy to use application. The system should not be too complex to engage in. It means that the functionality of the system will be basic. Since the purpose of the system is to provide users a friendly and easy to use learning environment, we will focus on the usability of the system rather than making it functional more than necessary. The system has a simple interface and familiar instructions to act instead of complex menus and various features. Thus, the users can spend time studying rather than struggling with complex features.

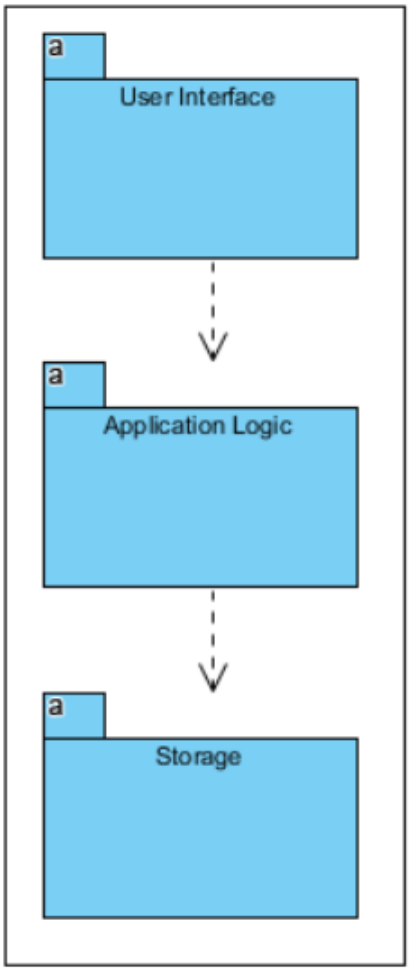
2. Software Architecture

2.1 Overview

Our main goal in this section is to decompose whole system into cohesive subsystems. In order to achieve this we will be using Three Layer Architectural style.

2.2 Subsystem Decomposition

In order to achieve the goals that are defined in 1st section, the system should be broken into three main pieces. This act of breaking into pieces is subsystem decomposition. Those so-called pieces are subsystems provide us which layers are associated with each other in which hierarchy, and also which classes are associated with each other in what terms. The classes with similar jobs belongs to the same subsystem components.



2.2.1 First Layer - User Interface

Our first layer is user interface, which includes User package in it. This layer is responsible for user interface creation and providing user friendliness and ease of use for the users. This layer scans the input from user's keyboard and mouse, and conveys this input to the second layer, which is application logic. Further, this layer is also responsible for differentiating interfaces for different roles of users, which are student and instructor.

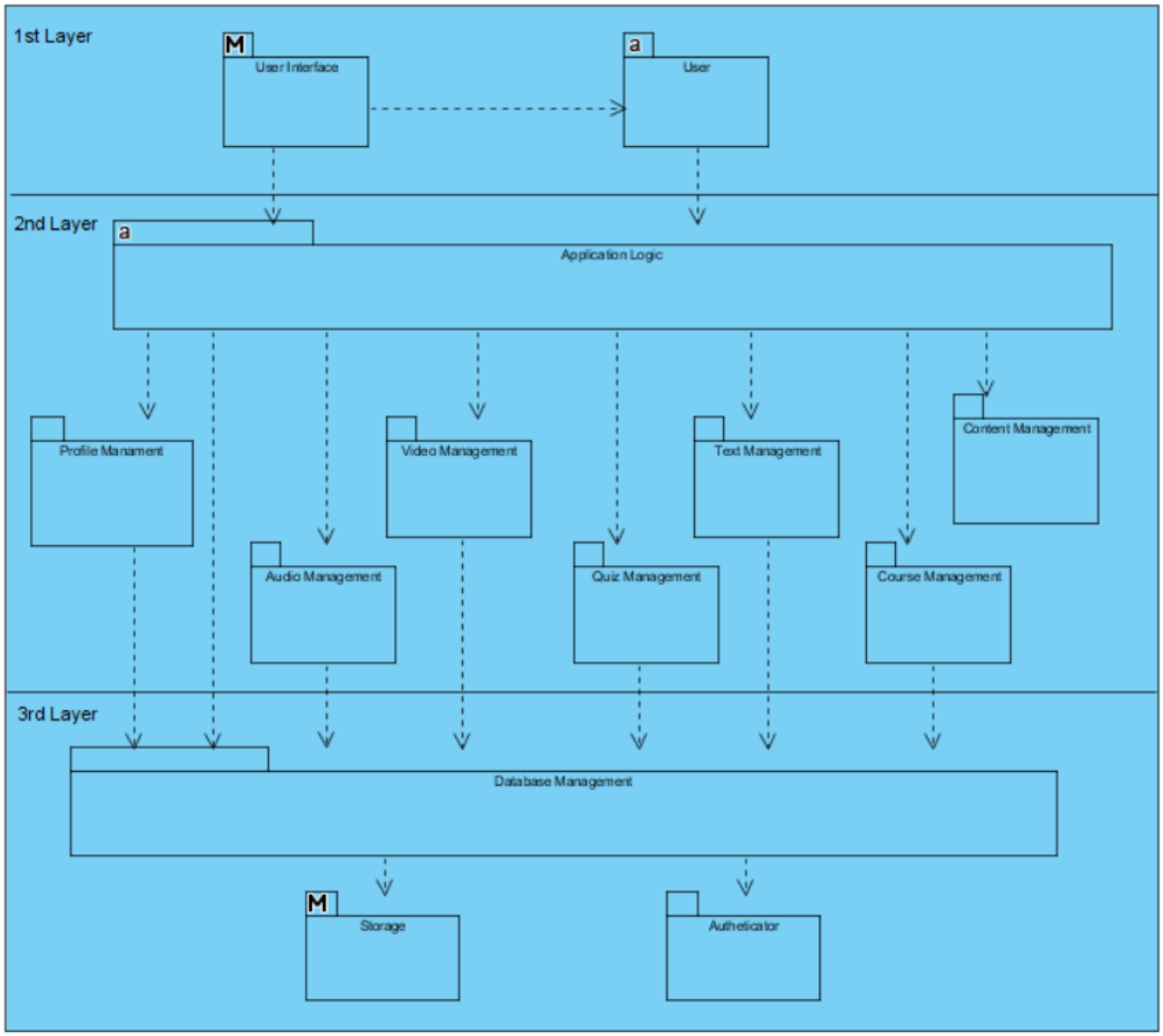
2.2.2 Second Layer - Application Logic

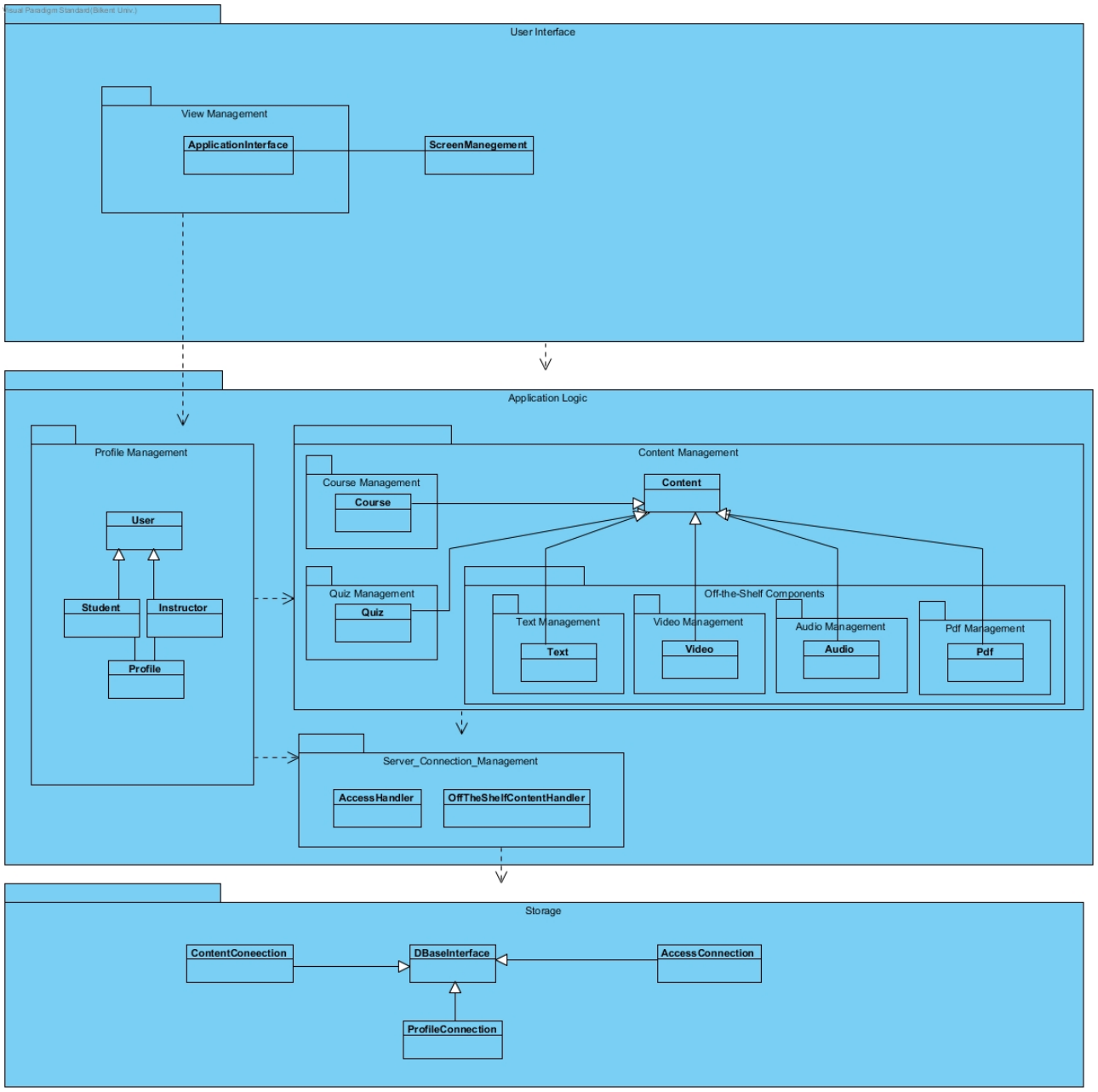
This layer's job is to do the desired job that is directed from the above layer, and also interpret the data coming from below layer, which is storage. This layer mostly deals with the main activities in the application. Inside this layer, there are profile management and content management packages. Profile management deals with the logic behind users' profiles, and content management deals with

the logic of contents of application related issues, such as quiz management and course management.

2.1.3 Third Layer - Storage

In this layer, there is a component that handles and simplifies the connection of database related issues. This layer's job is to provide course related information to the users, and handle the authentication-related issues of a user, according to his/her role. This one will insert and delete data to or from the database such as inserting a new quiz to the database, or deleting an existing one. It is also responsible for saving and updating users' personal information, course list, and other storage-related problems. Moreover, 3rd layer is also responsible with stopping the user interaction with application if their connection is lost, or any other system problems.





2.3 Hardware/Software Mapping

Mastering Bilkent will be implemented with Java programming language via JDK 8 and for GUI of our application JavaFX 8 will be intended to use. Thus, any computer having a platform which has JVM can be used to run our application.

For hardware, keyboard is needed for giving inputs to the application such as entering user profile information, entering course related informations or searching a specific course name. Mouse is

needed for also giving input such as clicking the wished course and wished course activity and a basic monitor is needed for interact with the system via GUI.

For storage, a database created with SQL. In order to bridge the gap between Java code and database, Java Database Connectivity (JDBC) API is needed.

2.4 Persistent Data Management

Mastering Bilkent has to store data related to users and courses. Thus, we can't use a temporary storage technique. A database implemented via SQL will be used.

*List of users with their roles,

*User's profile informations,

*Registered course list for each student,

*Created course list for each instructor,

*List of all courses with registered student list,

*Courses' documentation will be stored in our database.

2.5 Access Control and Security

Mastering Bilkent is intended to be an application which can be accessed by multiple users at the same time from different computers. Although an enterprise application need to satisfy this condition, because of the limited implementation time and CS 319 Course focuses on object design rather than database and internet connectivity we decide to implement our application to run on one computer for demonstration. Access through many computers will be leaved as future work at this point.

We will provide the security of the users' personal information. No user can access the other users' data. Also, nobody can modify the course's data other than it's own instructor. Course's and user's informations will be stored safely.

2.6 Boundary Conditions

Enter:

User shall enter the program by clicking the application icon. Application will be opened by .jar file.

Exit:

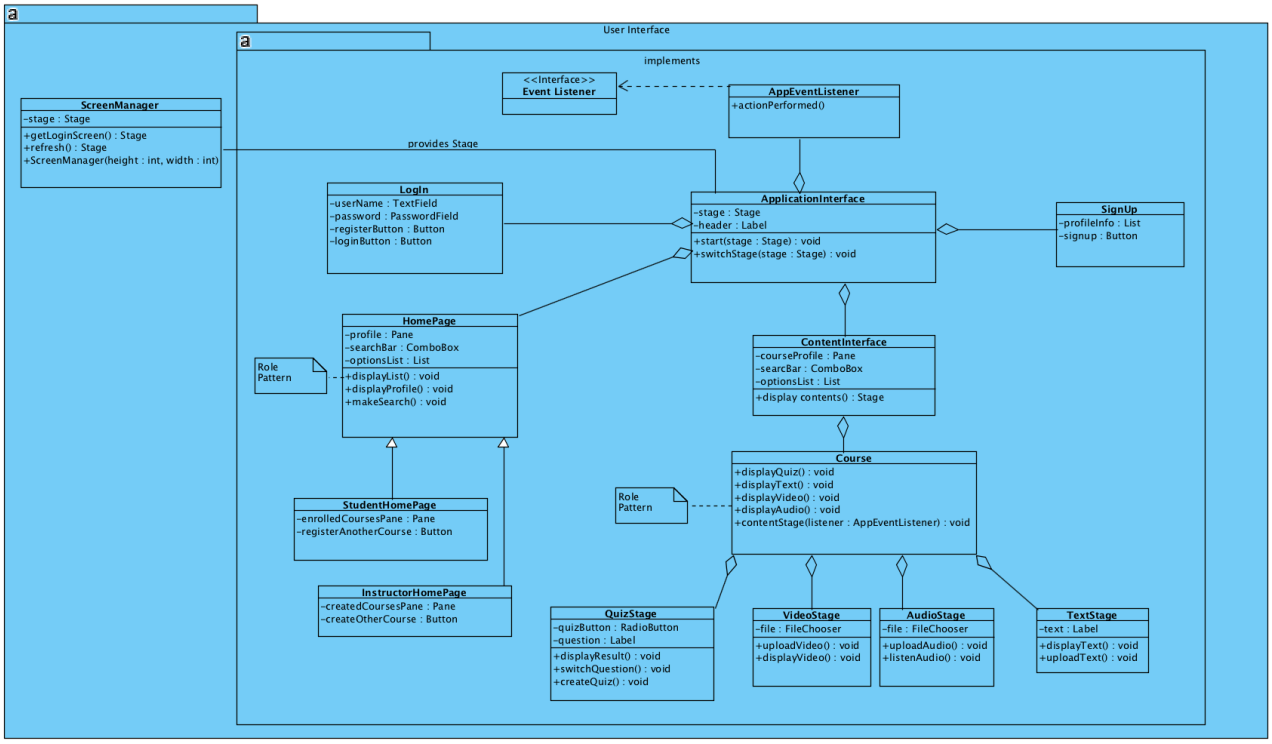
User shall exit with X button at the right top of the interface frame.

Error:

Until there won't be a problem related to database, users won't lose the informations and error screen will provide the user. However there is always a little risk to lose datas. If datas are lost, system will require from user to login again and courses must be created from beginning also. If connection to database is lost due to the user's system problem an error also will be shown.

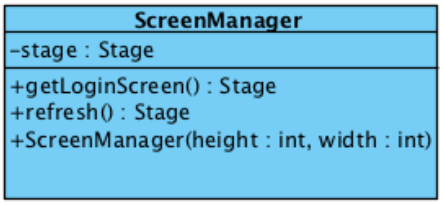
3. Subsystem Services

3.1 Graphical User Interface



User Interface Package handles with the graphical user interface by using JavaFX Library.

3.1.1 ScreenManager



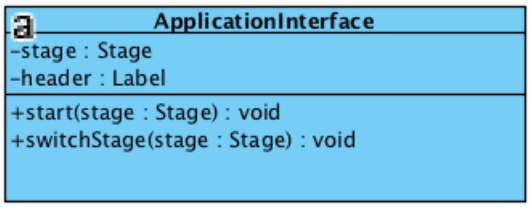
Attributes and Methods:

private Stage stage: This variable stands for frame object of the system.

public Stage getLoginScreen(): This operation takes user to Log In Screen.

public ScreenManager(int height, int width): This constructor that takes width and height for each creating determined stage.

3.1.2 Application Interface



Attributes and Methods:

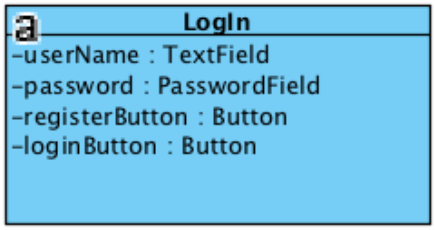
private Stage stage: This variable stands for stage object of the related classes.

private Label header: This variable stands for “Mastering Bilkent” header of stages.

public void start(Stage stage): This operation launches the stages.

public void switchStage(Stage stage): This operation switches the stages.

3.1.3 Login



Attributes :

private TextField userName: This variable stands for the text field which user will enter her/his username. Username will be e-mail address.

private PasswordField password: This variable stands for the password field which user will enter her/his password.

private Button registerButton: This button stands for switching context to register page.

private Button loginButton: This button stands for logging in to the system.

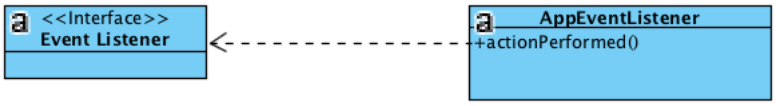
3.1.4 SignUp

Attributes :

private List profileInfo: This variable stands for the list view for user to enter registering information.

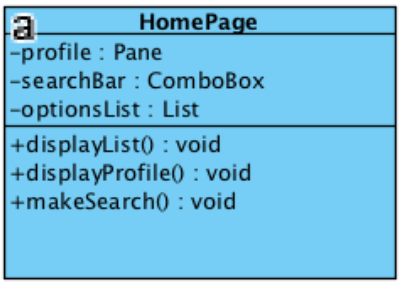
private Button signup: This button stands for signing up in to the system.

3.1.5 AppEventListener



AppEventListener class stands for user inputs of the system via the **actionPerformed()** method.

3.1.6 HomePage



Attributes and Methods:

private Pane profile: This variable stands for the pane for user's profile information.

private ComboBox searchBox: This variable stands for the search bar.

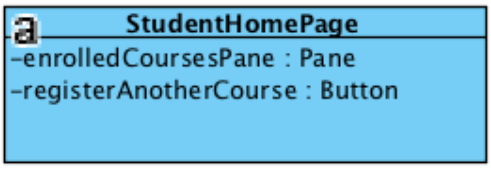
private List optionList: This variable stands for the different page options such as course, home, contents.

public void displayList(): This operation displays options mentioned above.

public void displayProfile(): This operation displays profile information.

public void makeSearch(): This operation is for making search by using searchBar variable.

3.1.7 StudentHomePage

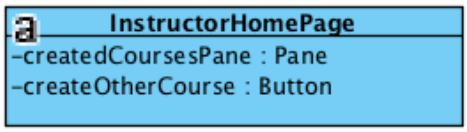


Attributes :

private Pane enrolledCoursesPane: This variable is for displaying pane of enrolled courses.

private Button registerAnotherCourse: This variable is for register course button.

3.1.8 InstructorHomePage

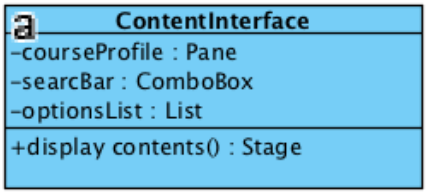


Attributes :

private Pane enrolledCoursesPane: This variable is for displaying pane of created courses.

private Button registerAnotherCourse: This variable is for create course button.

3.1.8 ContentInterface



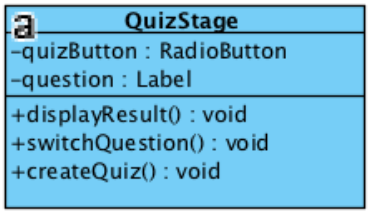
Attributes and Methods:

private Pane courseProfile: This variable stands for course related information(Creator Instructor, Course Name, Course Code).

private ComboBox searchBox: This variable stands for the search bar.

public Stage displayContents: This operation makes stages to display on the screen.

3.1.9 QuizStage



Attributes and Methods:

private RadioButton quizButton: This variable stands for answer buttons in quiz stage.

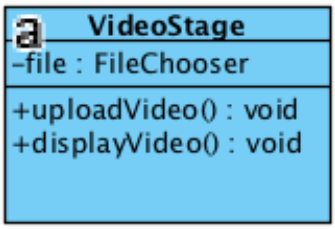
private Label question: This variable stands for quiz questions.

public void displayResult(): This method displays quiz results.

public void switchQuestion(): This method switches quiz questions.

public void createQuiz(): This method creates quizzes.

3.1.10 VideoStage



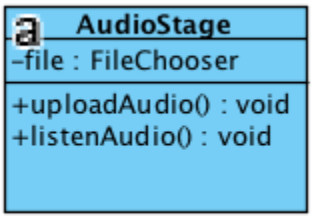
Attributes and Methods:

private fileChooser file: This variable stands for file chooser list for video upload.

public void uploadVideo(): This method is for uploading video to the system.

public void displayVideo(): This method is for watching video.

3.1.11 AudioStage



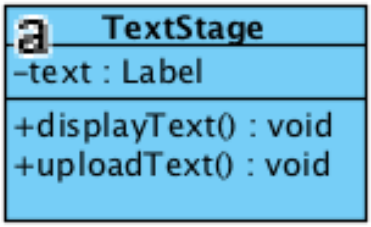
Attributes and Methods:

private fileChooser file: This variable stands for file chooser list for video upload.

public void uploadAudio(): This method is for uploading audio file to the system.

public void dlistenAudio(): This method is for listening an audio file.

3.1.12 TextStage



Attributes and Methods:

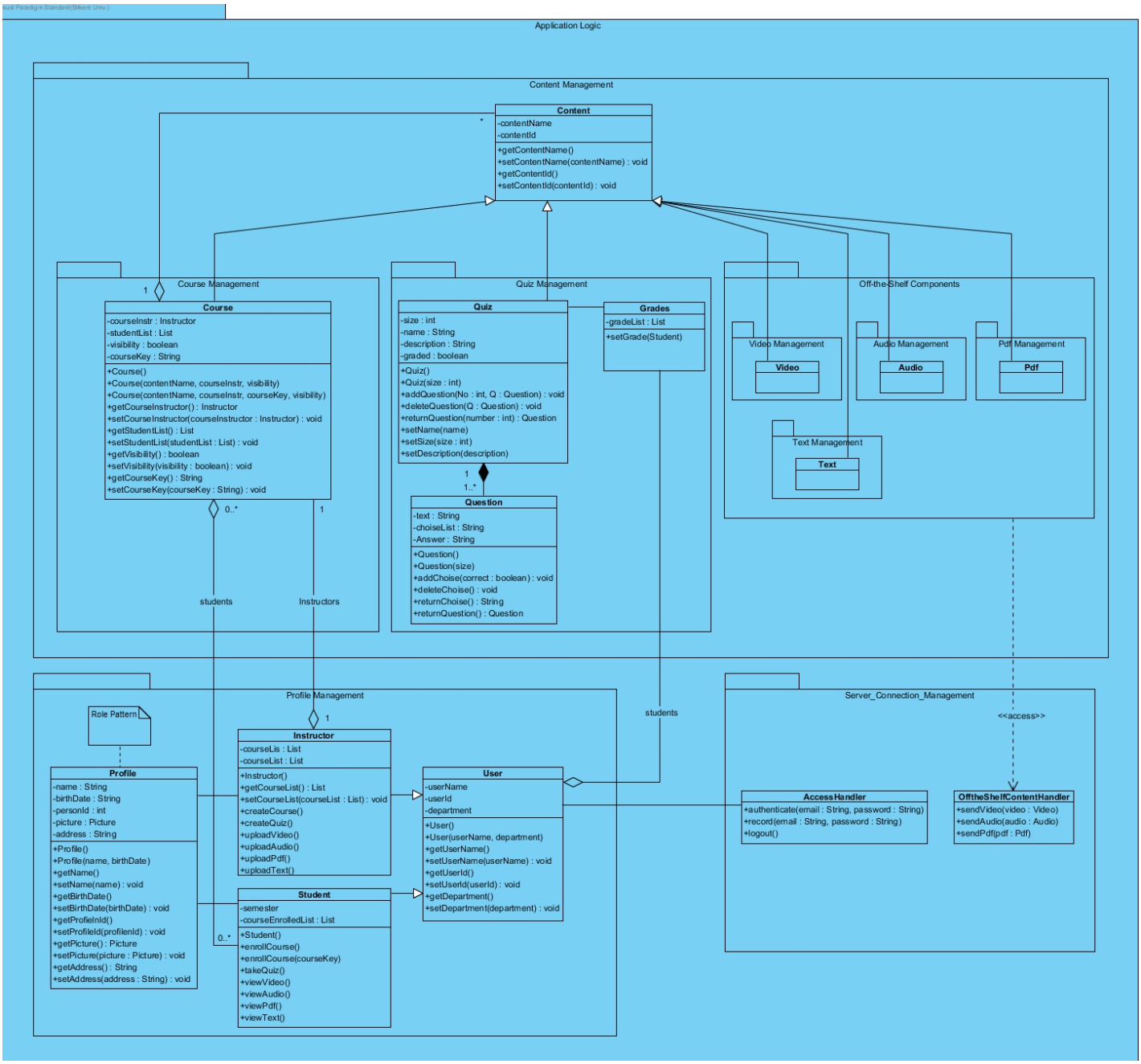
private Label Text: This variable stands for the text which will be displayed.

public void displayText(): This method is for displaying text.

public void uploadText(): This method is for uploading text.

3.2 Application Logic Subsystem

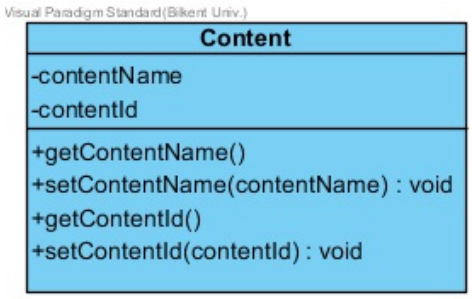
This layer Application Logic is also known as Business Logic and is responsible for the work done in the background in order to differentiate and manipulate the data relation to the business elements of the system. In the case of Mastering Bilkent they are the three main components of the system Content Management, Profile Management Subsystems and Server Communication Subsystem.



3.2.1 Content Management Subsystem

Content Management is the Subsystem of second layer where logic about the course materials like quizzes, video and audio files, pdf and text files are designed. It consists of Content Class, Course Management Subsystem, Quiz Management Subsystem, Off Shelf Components Subsystem.

3.2.1.1 Content Class



Content Class defines the component of any kind that the software can allow. A course, quiz, video, audio, text and pdf are all the contents the application can work with. And this list can be extended in the future. This allows *Mastering Bilkent* to manage its content easily by the component class from one place. Any addition in functionality will only affect component class which is kind of a façade approach to the problem.

Attributes and Methods:

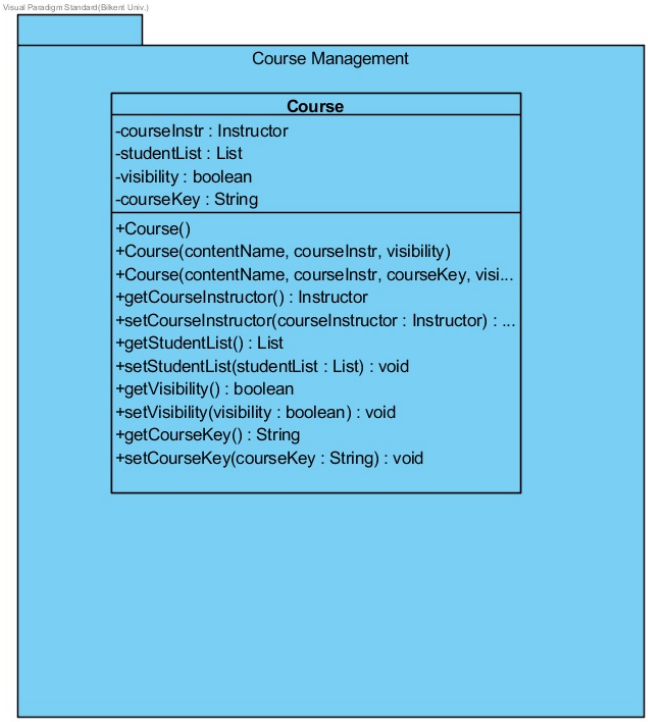
getContentName(): returns content name

setContentName(): changes content name

getContentId(): returns ID

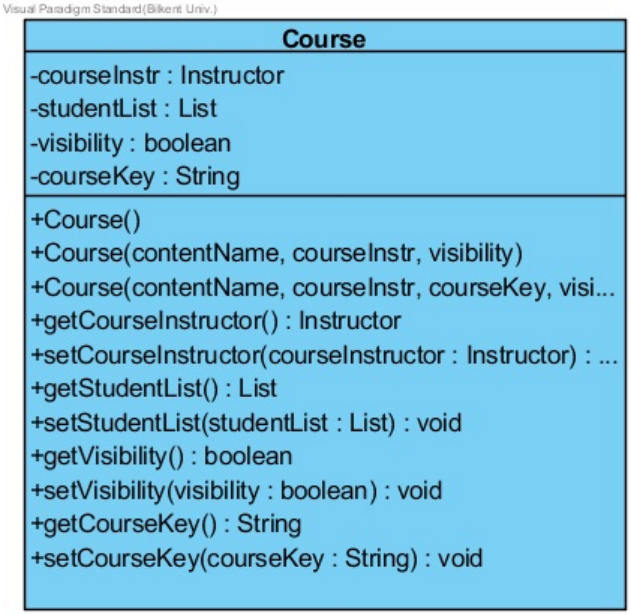
setContentId(): changes content ID

3.2.1.2 Course Management Subsystem



Course material subsystem is responsible for the logic that controls all functionality of courses. Although courses are components all other components of the application –apart from courses- are assign to a course and stored in the course. There is a “has a” relationship between course and components. Course class stores its Instructor, student list, visibility and course key.

3.2.1.2.1 Course Class



Attributes and Methods:

Course(): creates new course object with no specifications.

Course(contentName, courseInstr, visibility): creates new course object with name, instructor, visibility specification but no key.

Course(contentName, courseInstr, courseKey, visibility): creates new course object with name, instructor, key and visibility specification.

getCourseInstructor() : Instructor: return instructor of the course

setCourseInstructor(courseInstructor : Instructor) : void: changes instructor of a course

getStudentList() : List: return the list string the list of the student enrolled

setStudentList(studentList : List) : void: updates student list

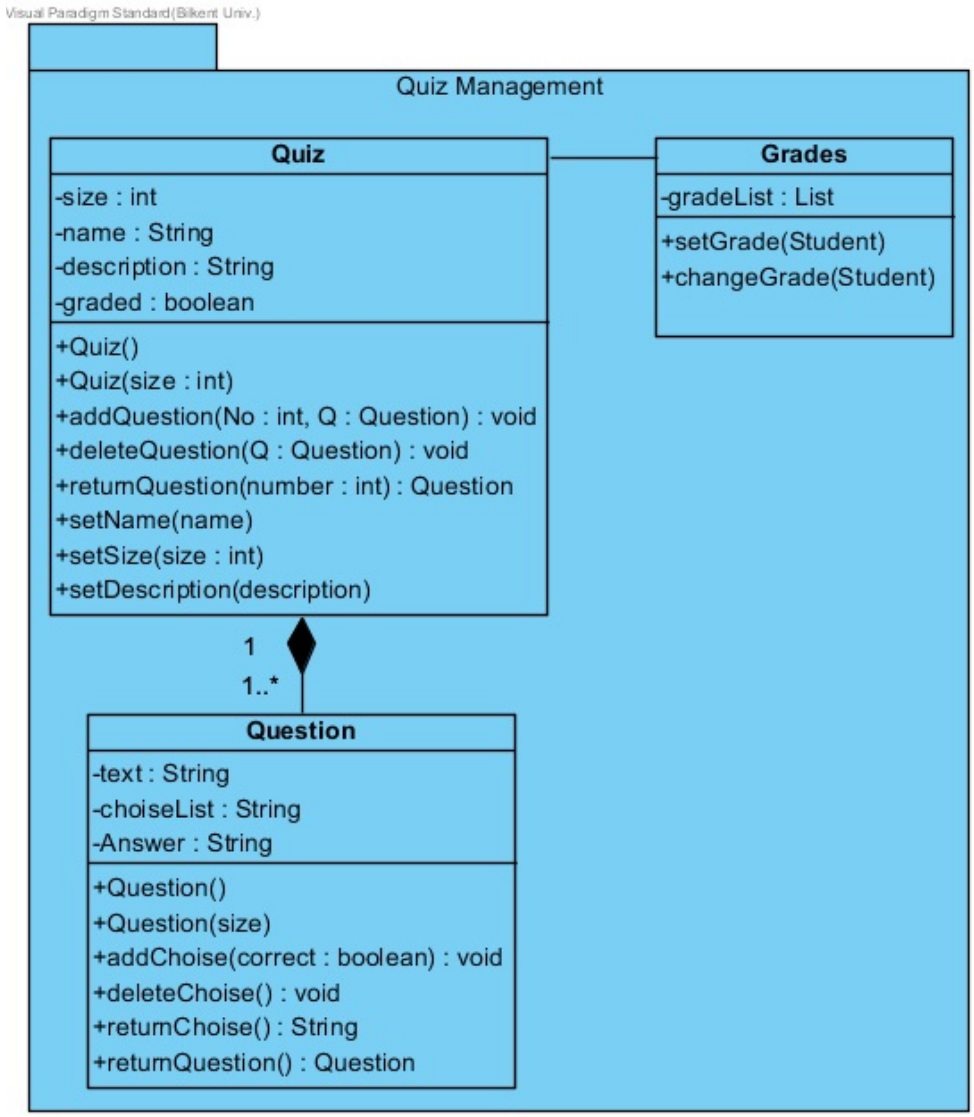
getVisibility() : boolean: returns the visibility value of the course

setVisibility(visibility : boolean) : void: changes visibility of a course

getCourseKey() : String: returns the key a the password if satisfied else returns no key

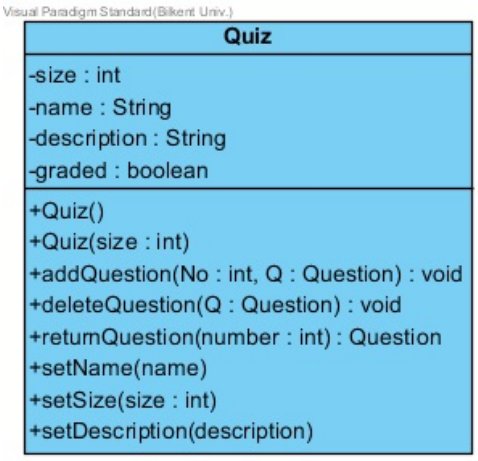
setCourseKey(courseKey : String) : void: changes the key of the course

3.2.1.3 Quiz Management Subsystem



Manages the all logic relating the quiz system of the application. It consists of Quiz Class, Question Class and Grades Class. An Instructor creates a quiz and adds it to his/her course and Students enrolled the course can attend the quiz.

3.2.1.3.1 Quiz Class



Defines functionalities of the quizzes.

Attributes and Methods:

Quiz(): default constructor

Quiz(size : int): constructor with value determining the number of questions in a quiz

addQuestion(No : int, Q : Question) : void: adds new question to the quiz

deleteQuestion(Q : Question) : void: deletes a question

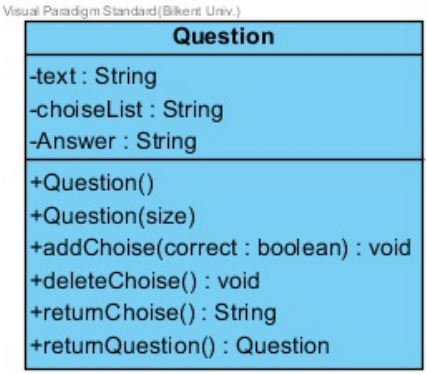
returnQuestion(number : int) : Question: returns the data of the question with given order in number

setName(name): changes the name of a quiz.

setSize(size : int): sets the maximum number of question possible in a quiz

setDescription(description): sets the description of the quiz

3.2.1.3.2 Question Class



Stores questions and allows them to be used.

Attributes and Methods:

Question(): default constructor

Question(size): constructor with the size variable defining number of choices

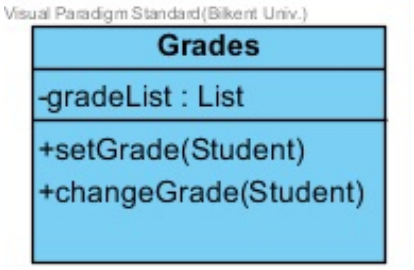
addChoise(correct : boolean) : void: adds new choice for question

deleteChoise() : void deletes a question

returnChoise() : String returns the choice text

returnQuestion() : Question: returns the question

3.2.1.3.3 Grades Class

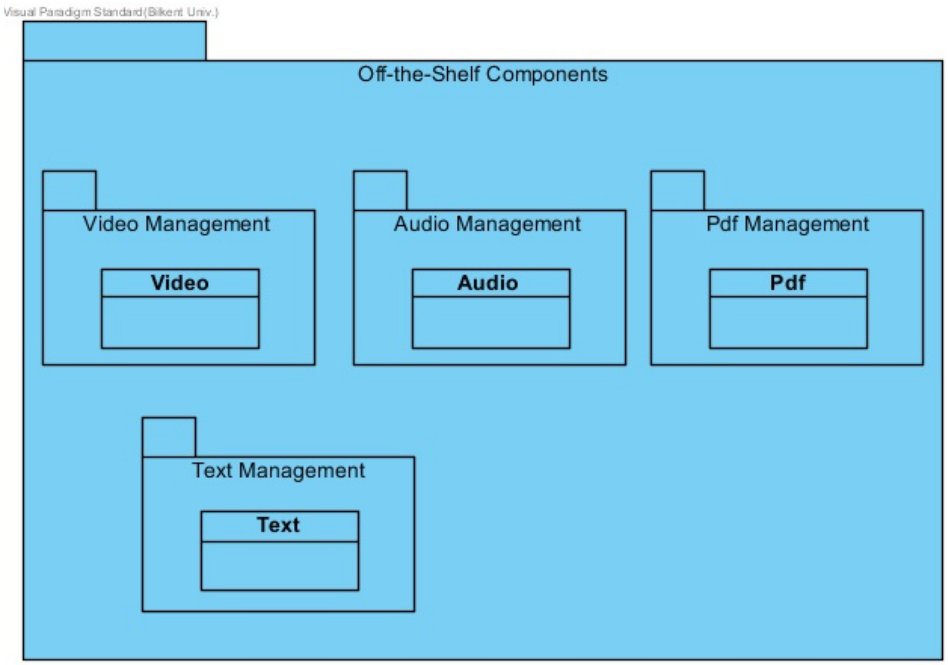


Attributes and Methods:

gradeList : List: stores grades of users attempting the course.

setGrade(Student): sets the quiz grade of a student user.

3.2.1.4 Off Shelf Components Subsystem



3.2.1.4.1 Video Management Subsystem

N/A - Will be determined by the system found online

3.2.1.4.2 Audio Management Subsystem

N/A - Will be determined by the system found online

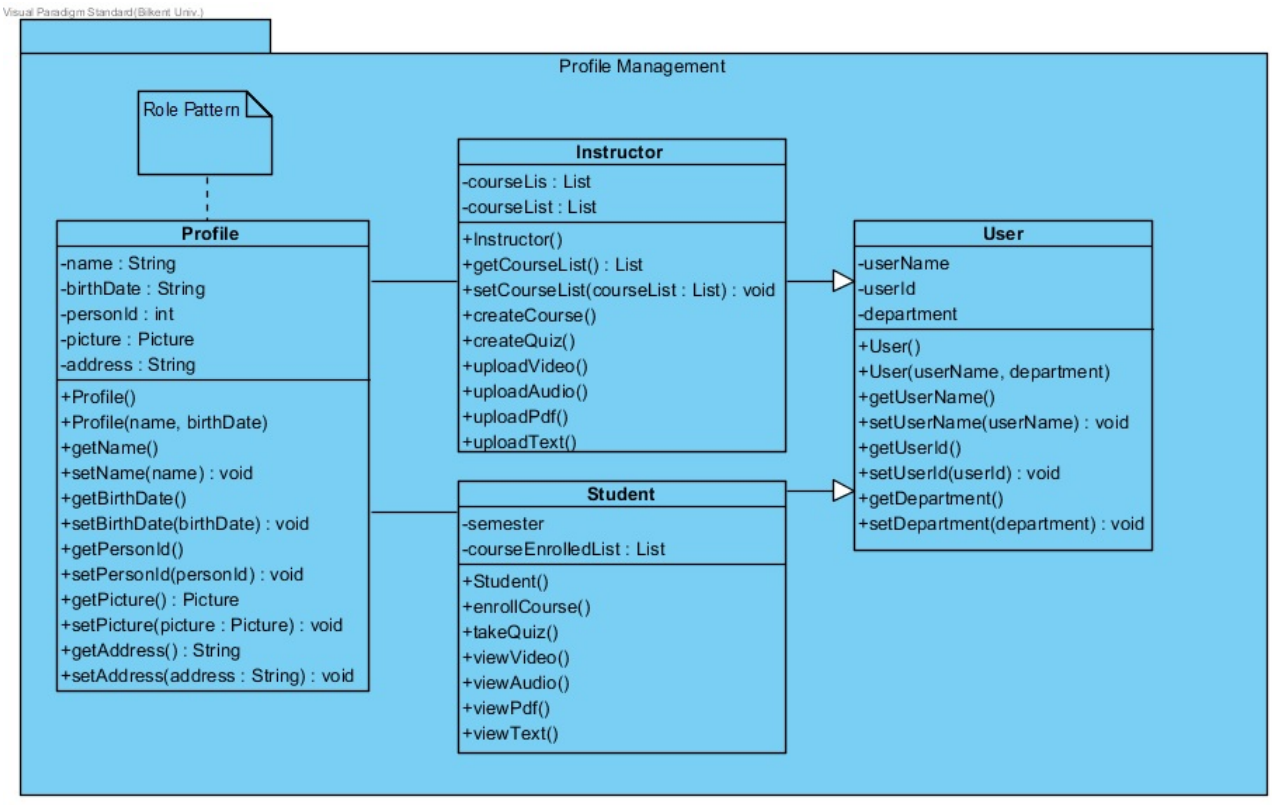
3.2.1.4.3 Pdf Management Subsystem

N/A - Will be determined by the system found online

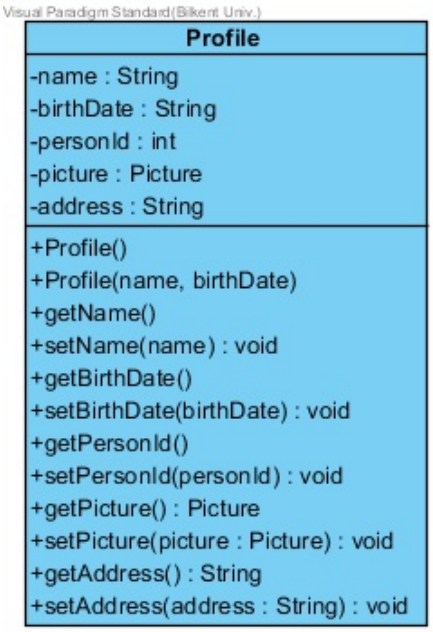
3.2.1.4.4 Text Management Subsystem

N/A - Will be determined by the system found online

3.2.2 Profile Management Subsystem



3.2.2.1 Profile Class



Attributes and Methods:

Profile(): default constructor

Profile(name, birthDate): contractor with personal information of name and birthdate

getName(): returns name of the profile owner

setName(name) : void: changes the name of the user

getBirthDate(): returns the birth data of user

setBirthDate(birthDate) : void: changes the birth date of the user

getProfileId(): returns Id of the profile

setProfileId(profileId) : void: changes the profile id

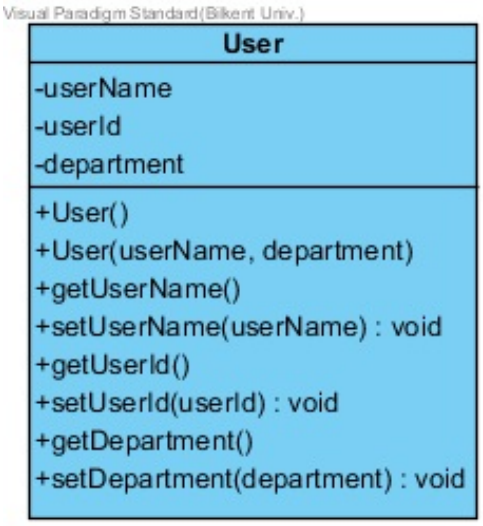
getPicture() : Picture: returns the picture of the user

setPicture(picture : Picture) : void: changes the user picture

getAddress() : String: returns the address info of a profile

setAddress(address : String) : void: changes address information

3.2.2.2 User Class



Attributes and Methods:

User(): default constructor

User(userName, department): creates new user object with a name and the department information

getUserName(): returns the user name

setUserName(userName) : void: changes the user name

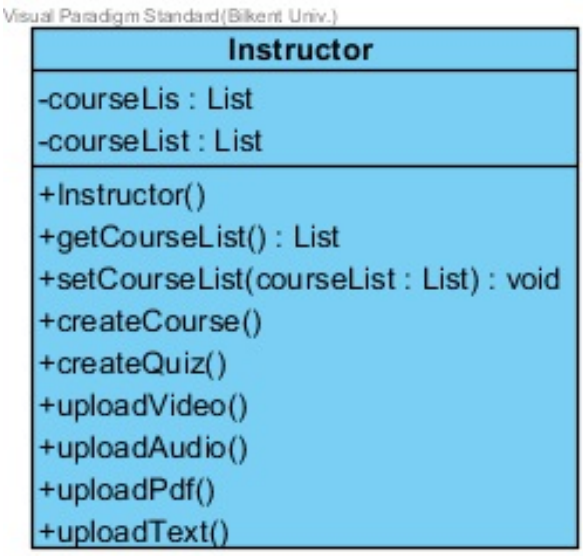
getUserId(): returns the user ID

setUserId(userId) : void: changes user ID

getDepartment(): returns user department information

setDepartment(department) : void: changes user information

3.2.2.3 Instructor Class



Instructor Class is the child class of User Class implements its methods. It is responsible for all the functionalities for an Instructor's use.

Attributes and Methods:

Instructor(): crates default Instructor object

getCourseList() : List: returns list of courses of and instructor

setCourseList(courseList : List) : void: manipulates course list

createCourse(): Allows the Instructor to create a new course

createQuiz(): Allows the Instructor to create a new quiz and updates course content list

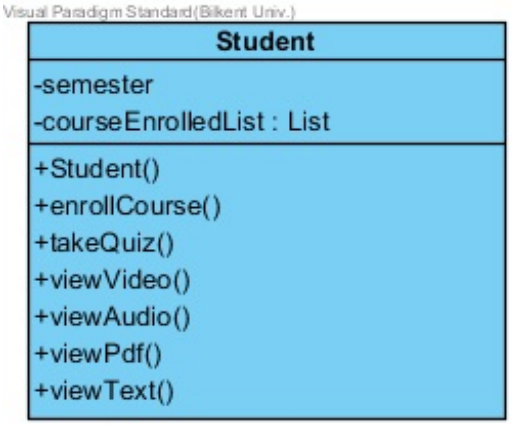
uploadVideo(): uploads a video to the server and updates course content list

uploadAudio(): uploads an audio to the server and updates course content list

uploadPdf(): uploads a pdf to the server and updates course content list

uploadText(): uploads a text to the server and updates course content list

3.2.2.4 Student Class



Student Class is the child class of User Class implements its methods. It is responsible for all the functionalities Students will use.

Attributes and Methods:

Student(): default constructor

enrollCourse(): Allows student to enroll a course without a key updates the course list of the student and the student list of the course:

enrollCourse(courseKey): Allows student to enroll a course without a key updates the course list of the student and the student list of the course:

takeQuiz(): calls the quiz data and allows Student to take a quiz

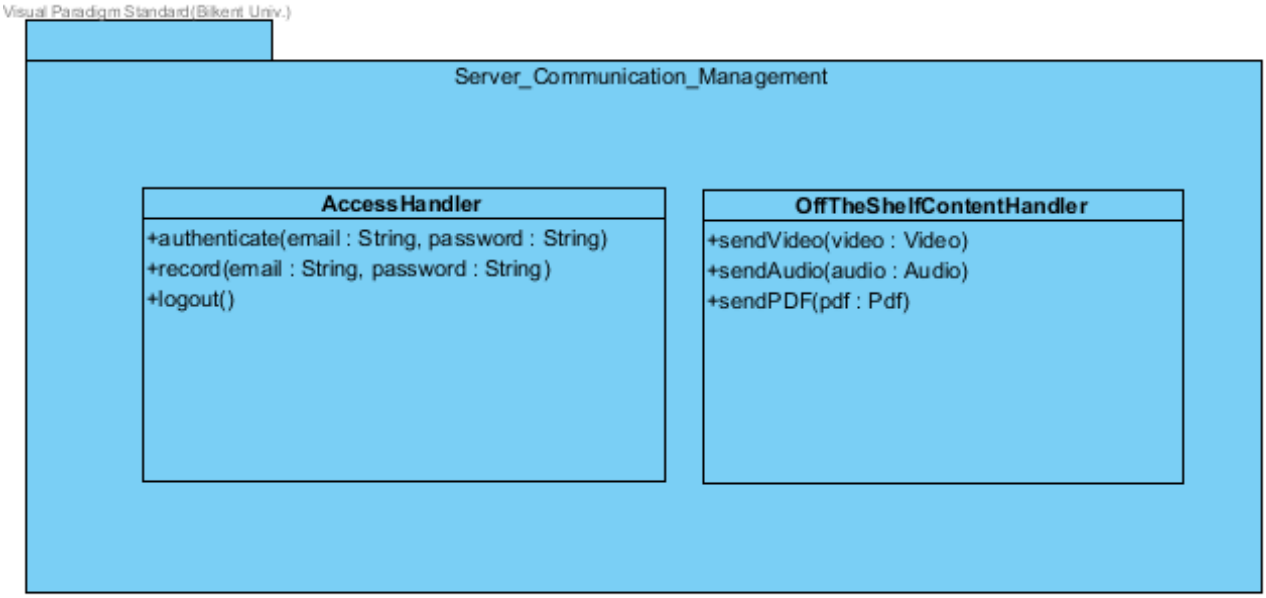
viewVideo(): calls the video from the server and allows student to view it

viewAudio(): calls the audio from the server and allows student to view it

viewPdf(): calls the pdf from the server and allows student to view it

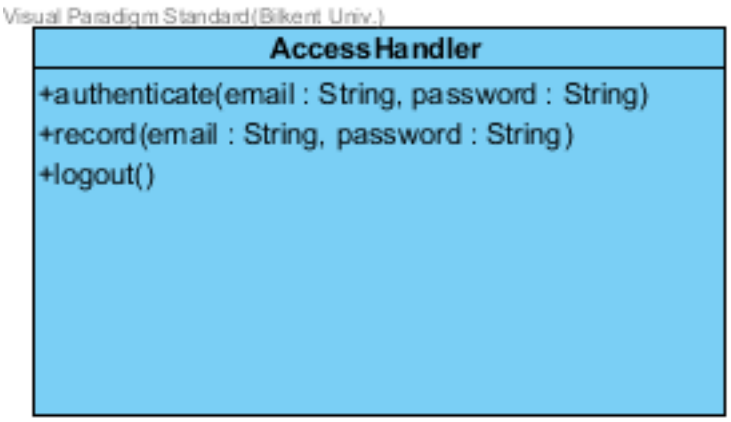
viewText(): calls the text from the server and allows student to view it

3.2.3 Server Communication Management



Server_Communication_Management package handles the database-user interactions for logging in and out. This package is also handles database-instructor interaction about uploading content. So it has two classes, one of them is AccessHandler for registration, log in and log out, and the other one is ContentHandler for uploading and deleting content to the system. Registration procedure checks whether the given e-mail, and password are valid or not. If these information are not valid, the registration system gives a warning to the user.Logging in procedure checks whether the entered information satisfies a user entity on database. For correct information, the AccessHandler loads all the data of the user from the database.

3.2.3.1 AccessHandler Class



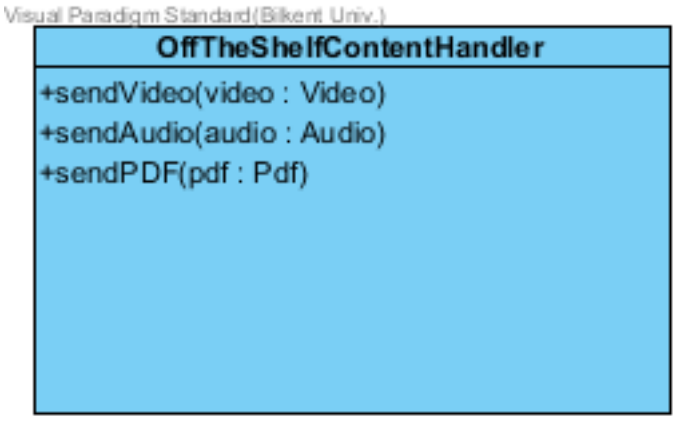
Attributes and Methods:

public int authenticate(String email, String password): This method's responsibility is to log the user in the system. It will use a connection to the database. It will return the user's ID, retrieved from the database, if the entered e-mail and passwords are valid. If the entered values are not valid then this method will return -1.

public void record(String email, String password): This method's responsibility is to record a new user to the database. This method gets the email and password inputs, and record this data in the database. Also, this will authenticate the newly created User to the system.

public boolean logout(): This method will return true if the user disconnected successfully, and false otherwise.

3.2.3.2 OffTheShelfContentHandler Class



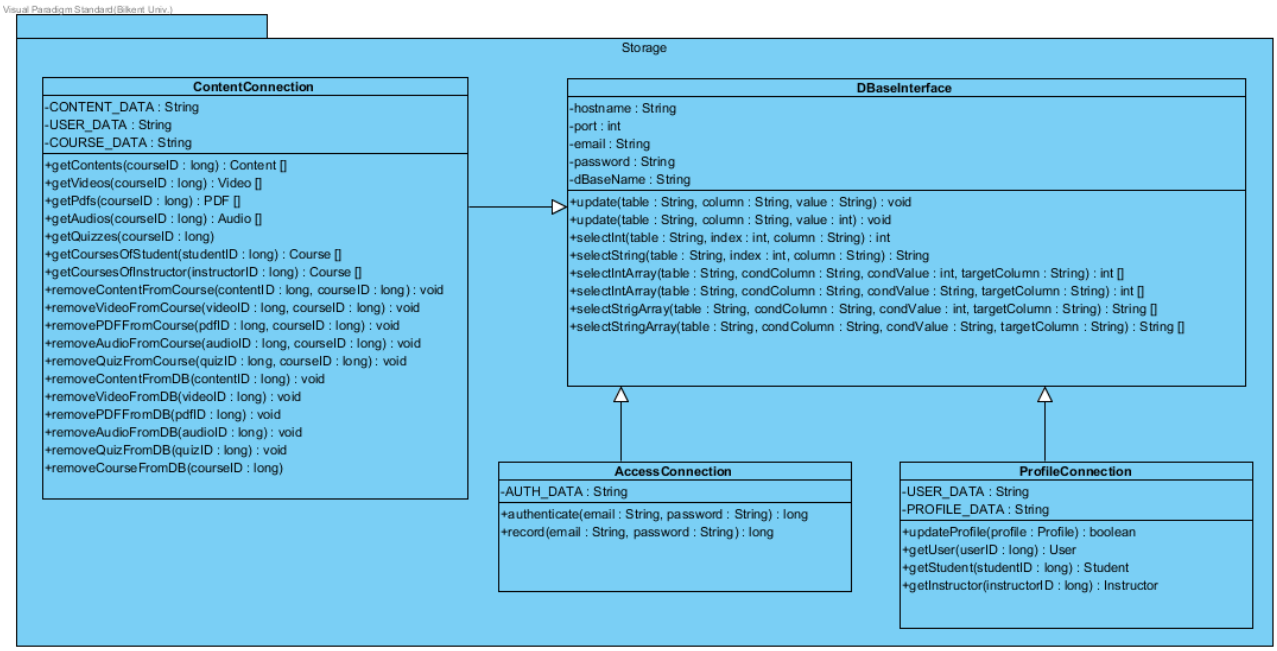
Attributes and Methods:

public void sendVideo(Video video): Sends a video to the database.

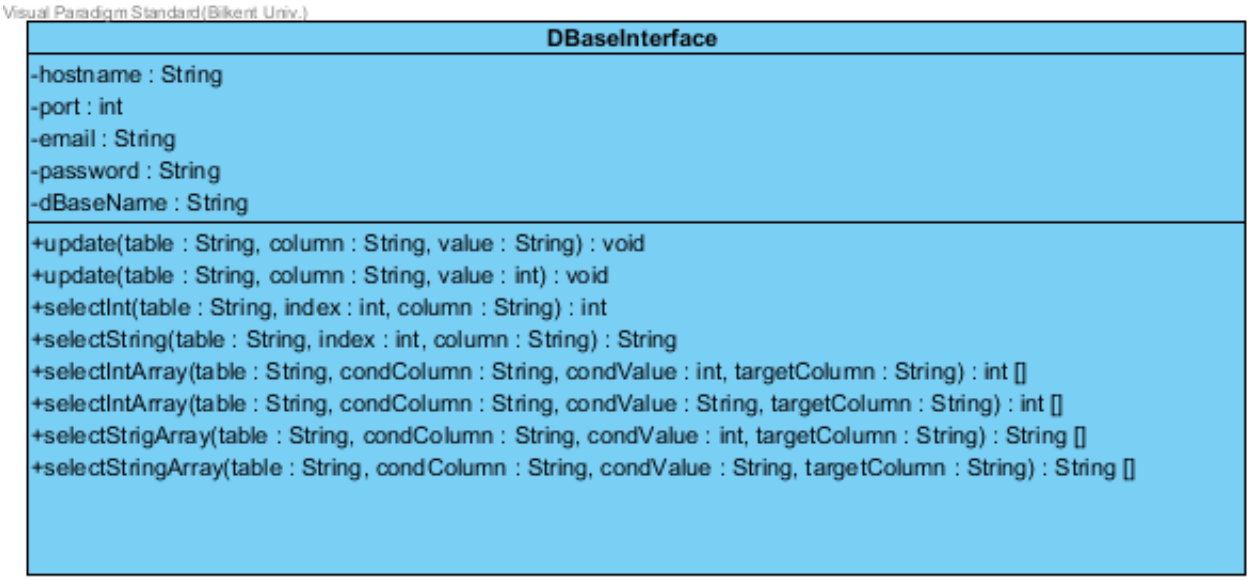
public void sendAudio(Audio audio): Sends an audio file to the database.

public void sendPDF(Pdf pdf): Sends a pdf file to the database.

3.3 Storage Subsystem



3.3.1 DBaseInterface Class



Attributes and Methods:

private String hostname: Host name of the database that will be used during connection.

private int port: Port that will be used during connection.

private String email: E-mail that will be used during connection.

private String password: Password of the database user.

private String dBaseName: The name of the database that will be used to store all the tables.

public void update(String table, String column, String value): Updates a table with the given string values.

public void update(String table, String column, int value): Updates a table with the given integer values.

public int selectInt(String table, int index, String column): Returns an integer from the database according to given address.

public String selectString(String table, int index, String column): Returns a string from the database according to given address.

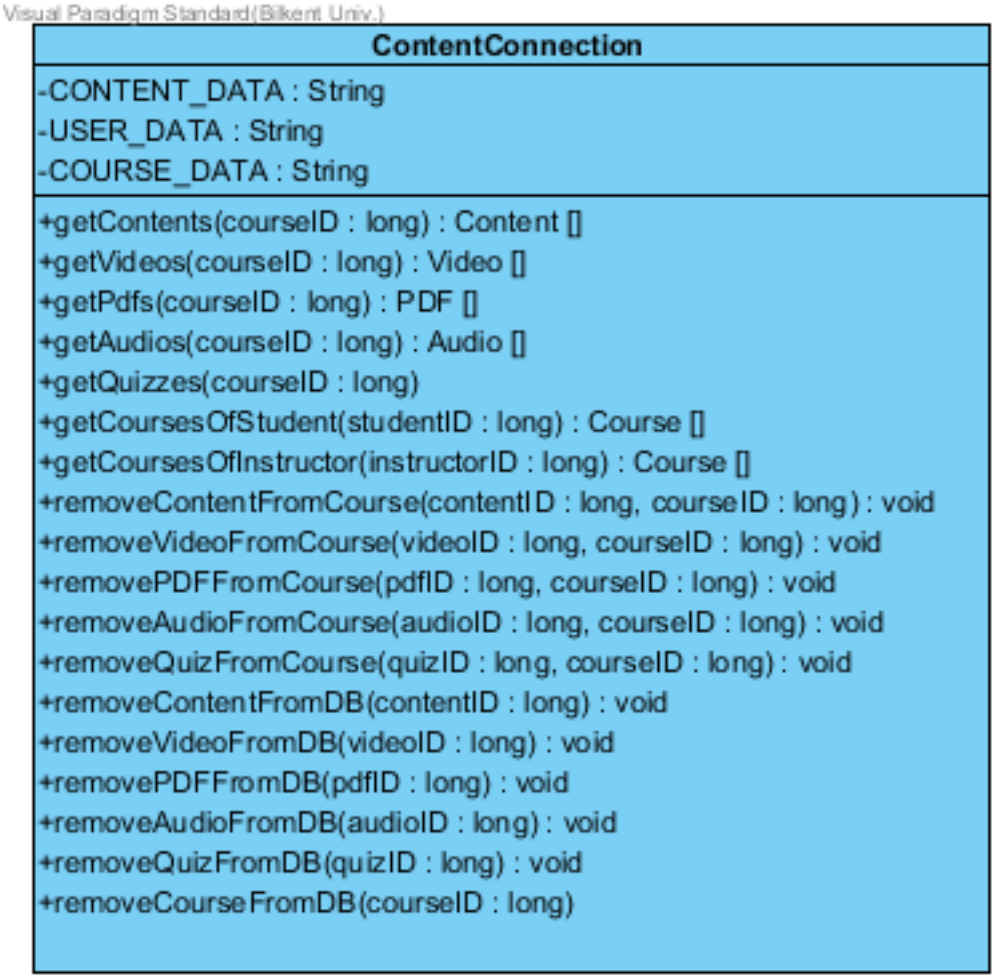
public int[] selectIntArray(String table, String condColumn, int condValue, String targetColumn): Returns an integer array from the database, which satisfies the given conditions.

public int[] selectIntArray(String table, String condColumn, String condValue, String targetColumn): Returns an integer array from the database which satisfies given conditions.

public String[] selectStringArray(String table, String condColumn, int condValue, String targetColumn): Returns a string array from the database which satisfies given conditions.

public String[] selectStringArray(String table, String condColumn, String condValue, String targetColumn): Returns a string array from the database which satisfies given conditions

3.3.2 ContentConnection Class



Attributes and Methods:

private String CONTENT_DATA: This attribute holds the name of the content table.

private String USER_DATA: This attribute holds the name of the user table.

private String COURSE_DATA: This attribute holds he name of the course table.

public Content[] getContents(long courseID): Returns the contents of the course with given course ID.

public Video[] getVideos(long courseID): Returns the array of videos in the given course ID.

public PDF[] getPdfs(long courseID): Returns the array of pdf files in the given course ID.

public Auido[] getAudios(long courseID): Returns the array of audio files in the given course ID.

public Quiz[] getQuizzes(long courseID): Returns the array of quizzes in the given course ID.

public Course[] getCoursesOfStudent(long studentID): Returns the array of courses taken by given student.

public Course[] getCoursesOfInstructor(long instructorID): Returns the array of courses given by instructor.

public void removeContentFromCourse(long contentID, long courseID): Removes a content from given course.

public void removeVideoFromCourse(long videoID, long courseID): Removes a video from given course.

public void removePDFFromCourse(long pdfID, long courseID): Removes a pdf from given course.

public void removeAudioFromCourse(long audioID, long courseID): Removes an audio file from given course.

public void removeQuizFromCourse(long quizID, long courseID): Removes a quiz from given course.

public void removeContentFromDB(long contentID): Removes a content from database completely.

public void removeVideoFromDB(long videoID): Removes a video from database completely.

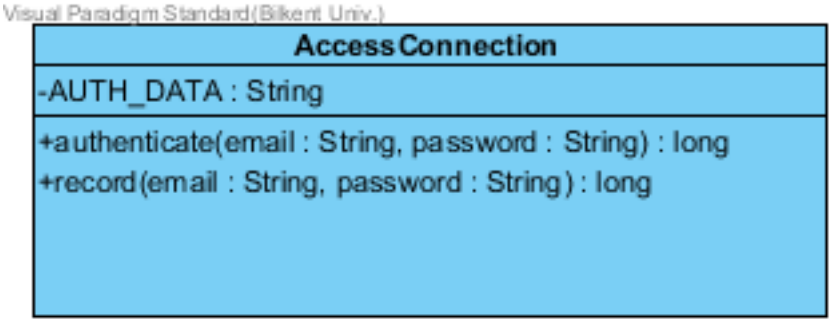
public void removePDFFromDB(long pdfID): Removes a pdf file from database completely.

public void removeAudioFromDB(long audioID): Removes an audio file from database completely.

public void removeQuizFromDB(long quizID): Removes a quiz from database completely.

public void removeCourseFromDB(long courseID): Removes a course from database completely.

3.3.3 AccessConnection Class



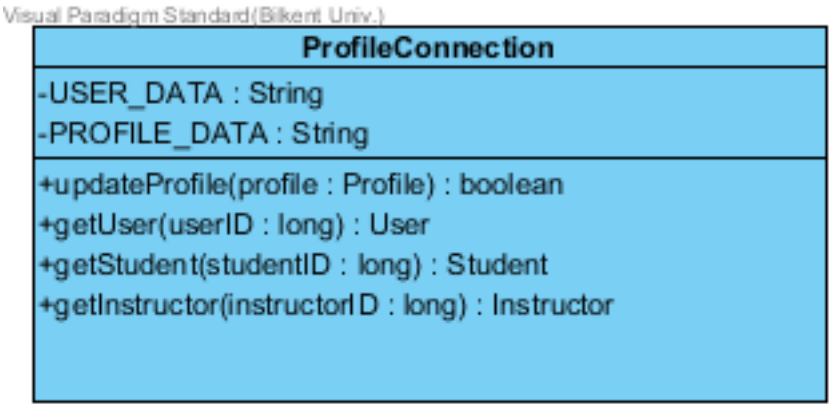
Attributes and Methods:

private String AUTH_DATA: This variable holds the name of the user in the database.

private long authenticate(String email, String password): Returns the unique ID of user with given email and password. If no user is found, this method will return -1.

private long record(String email, String password): Adds a new row to the User table according to the given values, and returns the index number of the new row in user table.

3.3.4 ProfileConnection Class



Attributes and Methods:

private String USER_DATA: This variable holds the name of the user in the database.

private String PROFILE_DATA: This variable holds the name of profile table in the database.

public boolean updateProfile(Profile profile): Updates the values in the database with the values given in Profile object.

public User getUser(long userID): Gets the values of given index in the user table and returns a user object with that values.

public Student getStudent(long studentID): Gets the values of given index in the student table and returns a student object with that values.

public Instructor getInstructor(long instructorID): Gets the values of given index in the instructor table and returns a instructor object with that values.