Bilkent University

Department of Computer Engineering

Senior Design Project

Automated Attendance Taking System (AATS)

Low Level Design Report

Alba Mustafaj, Argert Boja, Ndriçim Rrapi, Rubin Daija

Supervisor: Selim Aksoy
Jury Members: Ibrahim Korpeoglu and Hamdi Dibeklioglu

February 18, 2019

Contents

Low Level Design Report

Automated Attendance Taking System (AATS)

1. Introduction

The best way to understand a topic is by being involved in it. This is one of the main reasons why in universities participation in lectures is highly advised and sometimes even mandatory. However, occasionally student tend to skip their lectures but they still want to be counted as present, and for this they use several ways. This fact is widely known even by the professors, nevertheless, it is sometimes impossible to avoid it. Despite the alternative ways that can be used for checking whether all signatures are one-to-one, this can lead to inefficient usage of lecture time.

As we all might have experienced, the traditional way of checking for attendance is by passing a paper which contains all student names so that they can sign for themselves. This method of taking attendance is yet another reason for irregularities in this system. This is because sometimes the professors might forget to bring the attendance paper, or the students might forget to return the attendance paper. Furthermore, some students might come a few minutes late and they forget to sign the attendance.

This project aims to improve the methods of attendance taking, which will lead to less responsibilities for instructors and more accuracy for students. By using face recognition, we will be able to identify all the students present in class. Furthermore, by conducting several checks through the class hour there will be no possibility of cheating. Both students and instructors will be able to check the results on an application that will be build. Even if there is any inaccuracy, the students will be able to still be counted as present by letting their professor know that they were present in the taken picture. In this way, the new data will be collected and the system will be trained again in order to adapt to individuals changing their look.

This report aims to provide an overview of the low-level architecture and design of the project. Initially, design trade-offs are described and analyzed. Furthermore, the documentation guidelines are listed, followed by a description of the engineering standards. The second section focuses on the constituting packages and detailed description of the corresponding classes, interfaces and functionalities.

1.1 Design Trade-Offs

1.1.1 Cost vs. Accuracy

In order to have accurate results it is necessary to have a proper camera and take pictures from different angles. Thus, there is a cost-accuracy trade-off to be considered. Having 2 cameras increases the possibility of taking more accurate pictures and extracting relevant information for each student. Furthermore, the cameras itself should be relatively good in capturing details. Hence the more cameras the more information the system has. Furthermore the better and higher quality cameras cost more. Thus, there is a cost to accuracy tradeoff, whereby a middle ground needs to be found such that it does not impact economically the viability of this system.

1.1.2 Speed vs. Cost

Another area where cost has an impact is at speed. Since better hardware costs more, and the goal of this system is to be viable to implement, it is of outmost importance that the system can be utilized with lower tier hardware. Furthermore another goal of this system is scalability which necessitates the increase in use of the hardware, which in turn increases the cost overall. Thus there needs to be a tradeoff between speed of the system in a singular manner as well as in the whole scale, such that it is feasible in the long run.

1.2 Interface Documentation Guidelines

Class, variable and method names in the documentation follow the Camel-case format. Classes, variables, methods follow the respective hierarchy convention as below:

ClassName	
Class Description	
Attributes	
Attribute type Attribute name	
Methods	
Method name (parameters): return type	Method description

1.3 Engineering Standards

The reports utilize UML design principles [citation] for describing diagrams, interfaces, use cases and scenarios, subsystems, hardware and software components. Utilizing UML (a commonly used standard) makes it easier to represent the system, its components and functionalities.

1.4 Definitions, acronyms, and abbreviations

AATS – Automated Attendance Taking System

UML – Unified Modeling Language

UI – User Interface

HTTP - HyperText Transfer Protocol

FTP – File Transfer Protocol

JDBC – Java Database Connectivity

API: Application Programming Interface

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

Server: Part of system responsible for logical operations, scheduling, and data management

Client: Part of the system for user interaction

2. Packages

The system will include x packages: User Interface and Controller, Face Recognition, Database and Communication Package User Interface and Controller Package will be inside of the Client Subsystem, whereas the Face Recognition, Database and Communication Package will be part of the Server Subsystem.

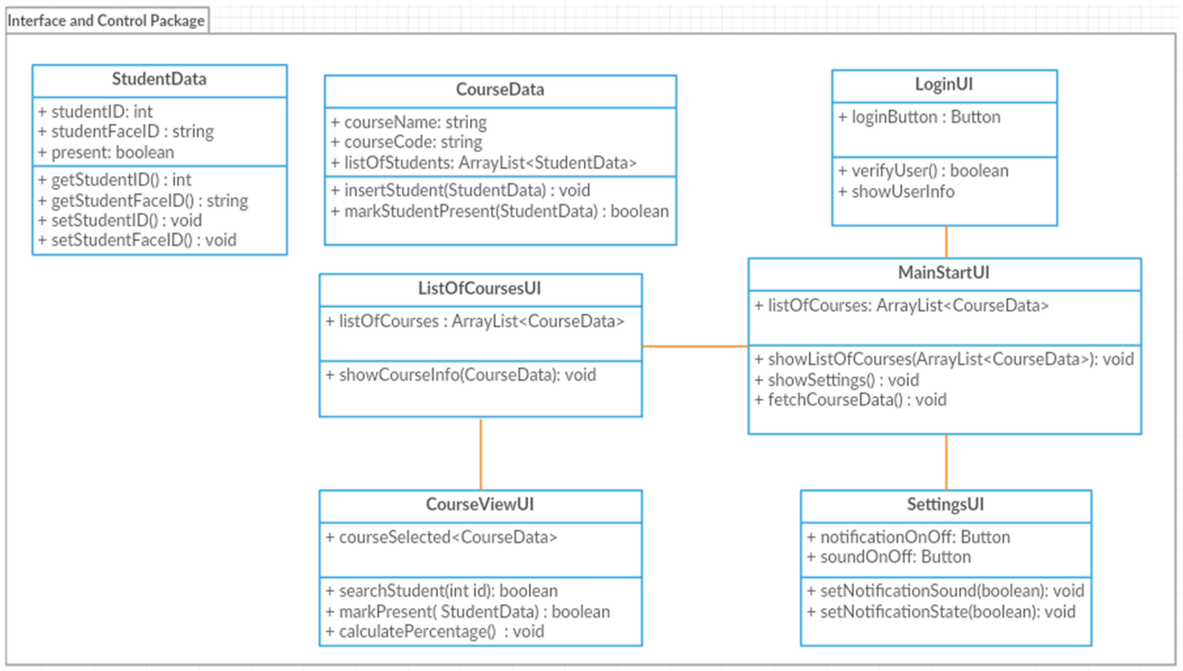
2.1 Client

The client subsystem will be composed of a single package named User Interface and Controller. Thus, the user interface and the controller will be dealt with within the same package being that these two parts happen to be tightly coupled within themselves being that our user interface and control system are mainly basic views and do not include a lot of control options as well. Thus separating them into two different packages would create redundant very small packages instead of a single compound one.

2.1.1 Interface and Control Package

This package deals with all of the operations on the client side of the application. Firstly the package manages to interpret all of the incoming data from the server into a proper interface for the users to view all of the necessary data. Next, the package also incorporates its ability to grant the user control options such as marking students present in class by the press of a button or accepting and rejecting students' attendance review requests.

The figure below shows the structure of the package including all of the involved classes and their main functions.



Interface and Control Package
StudentData
+ studentID: int
+ studentFaceID : string
+ present: boolean
+ getStudentID() : int
+ getStudentFaceID() : string
+ setStudentID() : void
+ setStudentFaceID() : void
CourseData
+ courseName: string
+ courseCode: string
+ listOfStudents: ArrayList<StudentData>
+ insertStudent(StudentData) : void
+ markStudentPresent(StudentData) : boolean
LoginUI
+ loginButton : Button
+ verifyUser() : boolean
+ showUserInfo
ListOfCoursesUI
+ listOfCourses : ArrayList<CourseData>
+ showCourseInfo(CourseData): void
MainStartUI
+ listOfCourses: ArrayList<CourseData>
+ showListOfCourses(ArrayList<CourseData>): void
+ showSettings() : void
+ fetchCourseData() : void
CourseViewUI
+ courseSelected<CourseData>
+ searchStudent(int id): boolean
+ markPresent(StudentData) : boolean
+ calculatePercentage() : void
SettingsUI
+ notificationOnOff: Button
+ soundOnOff: Button
+ setNotificationSound(boolean): void
+ setNotificationState(boolean): void

Figure 1 : Interface and Control Package Diagram

A brief explanation on each class included in this package is included below:

- **MainStartUI**
  This class shows the screen with the main profile data of the logged on user. From this screen the user can control the press of three buttons which lead them to the **ListOfCoursesUI**
  view class and ***SettingsUI*** view class.
- **LoginUI**
  This class is used to show the login screen. The user here has some control options by inputting their email and password in the related boxes. The class is responsible for checking whether the server approves the login request or not. The ***LoginUI*** class then redirects to the ***MainStartUI*** view class.
- **ListOfCoursesUI**
  This view class will show the profile data of the logged on user adjacent to a list of the courses this user is registered in as an instructor. The user has the control option to click a course. In such a scenario the user is redirected to the **CourseViewUI.**
- **CourseViewUI**
  This class will show the main course details in text including absences, date, present percentage and other relevant data. Adjacent to this data the user may also view a grid of pictures of all of the students present and not present marking them clearly from one another. Additionally the user may use the search bar to search for a particular student in case the grid is large. The user has the option to mark any students that were not recognized present by clicking on the relevant student picture on the grid or searching by name or id number and then then selecting the mark present option manually.
- **CourseData**
  This class will contain the data for a single course such as the name of the course, course code, list of the ids of the students of that course as well as a list of all of the urls of the cropped student face picture.
- **StudentData**
  This class will contain the main data for a single student such as the id of the student and the url of the students face image if there is any.
- **SettingsUI**
  This settings class is mainly used to manage notification settings like settings the sound on or off options.

2.2 Face Recognition Package

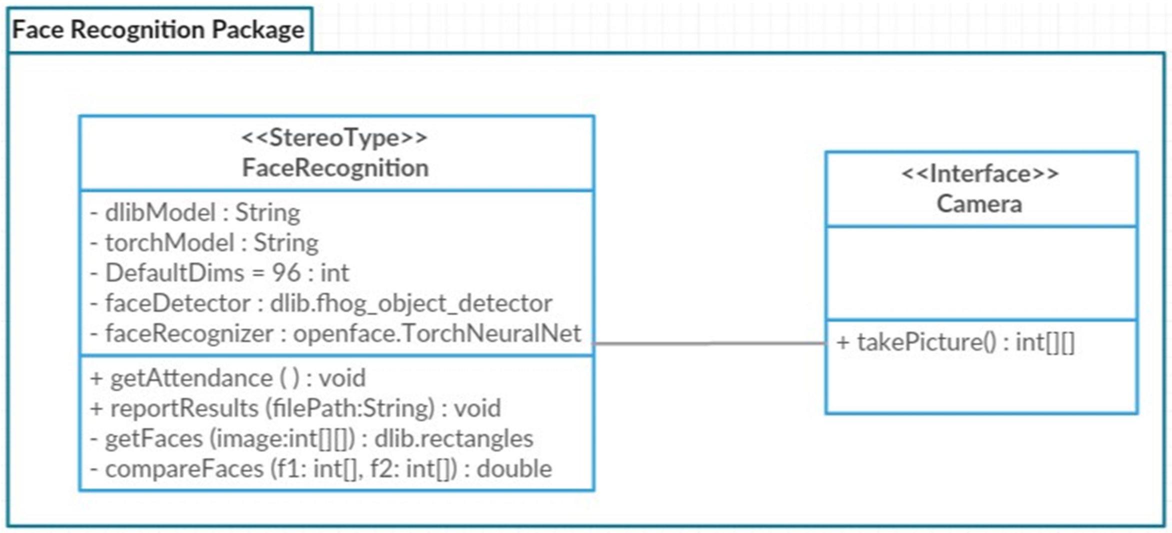

Face Recognition Package
<<StereoType>>
FaceRecognition
- dlibModel : String
- torchModel : String
- DefaultDims = 96 : int
- faceDetector : dlib.fhog_object_detector
- faceRecognizer : openface.TorchNeuralNet
+ getAttendance ( ) : void
+ reportResults (filePath:String) : void
- getFaces (image:int[][]) : dlib.rectangles
- compareFaces (f1: int[], f2: int[]) : double
<<Interface>>
Camera
+ takePicture() : int[][]

Figure 2: Face Recognition Package

A brief explanation on each class included in this package is included below:

- **FaceRecognition**
  This class will do the detection of the faces in a class room, the extraction of the feature vectors and then the comparison between different student vectors in order to identify individuals.

- **Camera**
  Is an interface since its implementation will be dependent on the camera unit used and the architecture where this system is deployed. This interface will have the takePicture() function which will be used by the face recognition class to identify individuals in the class.

2.3 Database Package

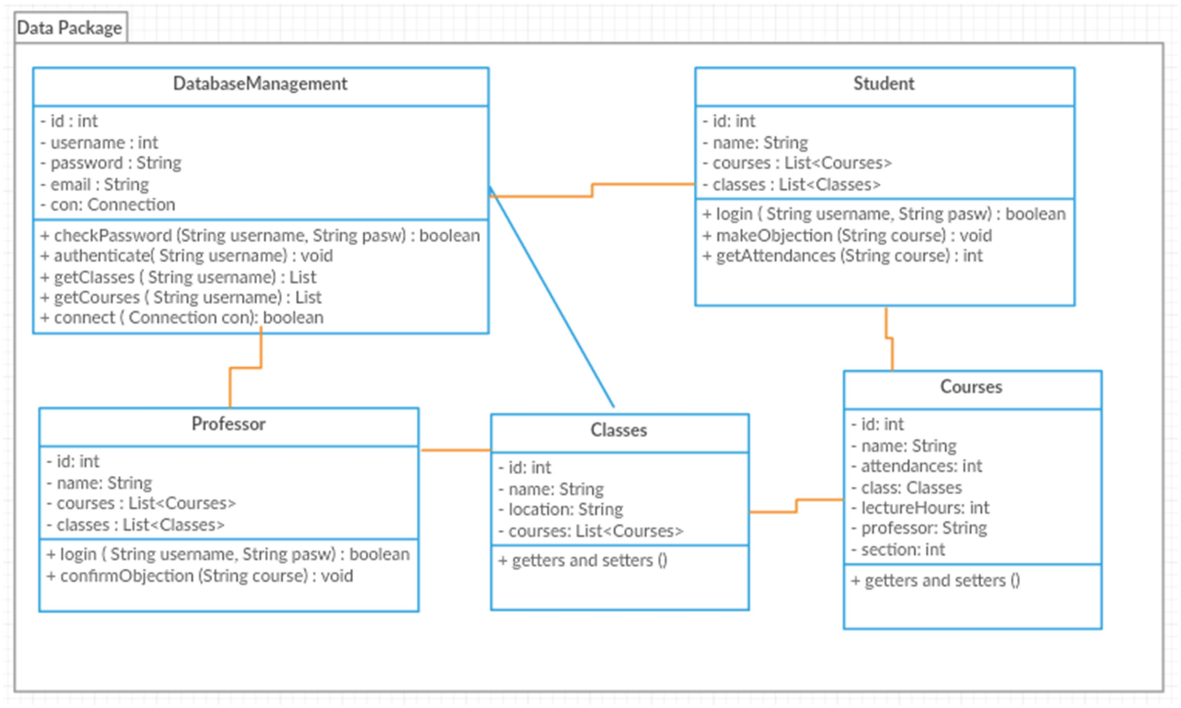

Data Package
DatabaseManagement
- id : int
- username : int
- password : String
- email : String
- con: Connection
+ checkPassword (String username, String pasw) : boolean
+ authenticate(String username) : void
+ getClasses (String username) : List
+ getCourses (String username) : List
+ connect (Connection con): boolean
Student
- id: int
- name: String
- courses : List<Courses>
- classes : List<Classes>
+ login (String username, String pasw) : boolean
+ makeObjection (String course) : void
+ getAttendances (String course) : int
Professor
- id: int
- name: String
- courses : List<Courses>
- classes : List<Classes>
+ login (String username, String pasw) : boolean
+ confirmObjection (String course) : void
Classes
- id: int
- name: String
- location: String
- courses: List<Courses>
+ getters and setters ()
Courses
- id: int
- name: String
- attendances: int
- class: Classes
- lectureHours: int
- professor: String
- section: int
+ getters and setters ()

Figure 3 Data Package

A brief explanation on each class included in this package is included below:

- **Student class**

This class is responsible for storing the student data. It will also serve as the class which will enable the student to log in the system. After logging in the student will be redirected to the main class. This process will be enabled through communication with the **DatabaseManagement class.** In this class there will also be stored information such as the classes and courses the student attends and also the attendance results. Through this class the student will also be able to make an objection in case his face has not been recognized by the system.

- **Professor class**

This class is responsible for storing the professor data. It will also serve as the class which will enable the professor to log in the system. After logging in the professor will be redirected to the main class. This process will be enabled through communication with the **DatabaseManagement class.** In this class there will also be stored information such as the classes and courses the professor attends. Furthermore, the professor will also be able to respond to an objection made by his/her students.

- **Classes class**

This class will store information of each class. It also has a list of **Courses** objects which will show which courses take place in this class. Also the location where this class is, is also stored in here.

- **Courses class**

This class will store information for which course. It will contain information regarding the section, number of lecture hours, the professor, the number of students and in which class this course takes place.

- **DatabaseManagement Class**

This is the main class of this package. This class will make possible the connection of the clients to the database in the server though the connection method. It will also enable students and professors to log in the system if their credentials are correct. Furthermore, through this class will be possible to populate the other data saving classes mentioned above though gettters and setters methods.

2.4 Communication Package

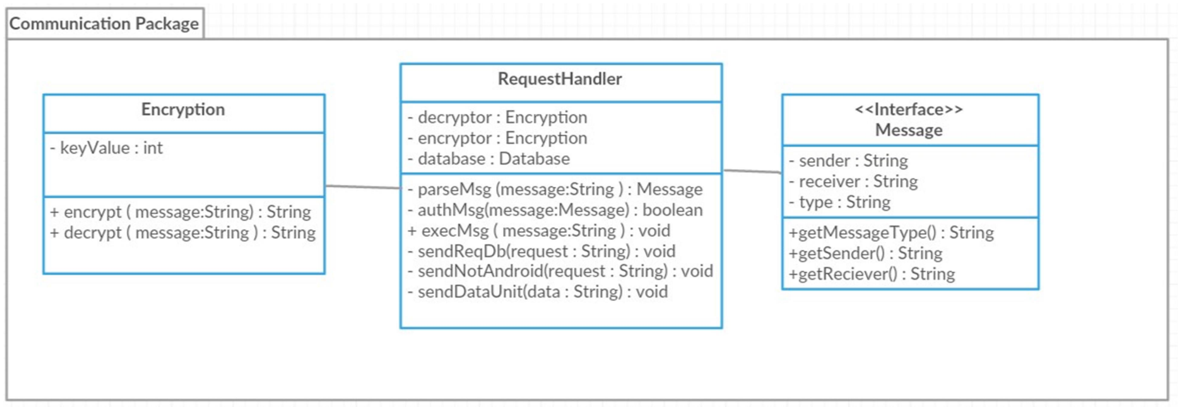

Communication Package
Encryption
- keyValue : int
+ encrypt (message:String) : String
+ decrypt (message:String) : String
RequestHandler
- decryptor : Encryption
- encryptor : Encryption
- database : Database
- parseMsg (message:String) : Message
- authMsg(message:Message) : boolean
+ execMsg (message:String) : void
- sendReqDb(request : String) : void
- sendNotAndroid(request : String) : void
- sendDataUnit(data : String) : void
<<Interface>>
Message
- sender : String
- receiver : String
- type : String
+getMessageType() : String
+getSender() : String
+getReciever() : String

Figure 4: Communication Package

A brief explanation on each class included in this package is included below:

- **Encryption**
  This class will deal with encrypting and decrypting the messages in order to make the communication process more secure.
- **RequestHandler**
  This class will be used for the handling of requests. It will decrypt the requests, authenticate them and then parse them and then execute them.

- **Message**
  This message interface will be used to implement the different message formats that will be used throughout the system.

3. Class Interfaces

3.1 Client Interface

3.1.1 Interface and Control Package Interface

class MainStartUI	
This class shows the screen with the main profile data of the logged on user and helps to redirect user to the list of courses screen and settings	
Attributes	
private ArrayList<CourseData> listOfCourses	
Methods	
showListOfCourses(ArrayList<CourseData> listOfCourses) : void	This method uses the parsed listOfCourses array and switches to the new activity ListOfCoursesUI
showSettings() : void	This method is used to redirect the user to the SettingsUI activity
fetchCourseData() : void	This method is used to retrieve the data of the users lecturing courses and store them into the listOfCourses arraylist.

class LoginUI	
This class shows the screen activity with the log in input boxes and the log in buttons. By using input from the user it enables them to log into the system.	
Attributes	
private Button loginButton	
Methods	
verifyUser() :boolean	This method fetches the user input from the input boxes and verifies if the entered credentials are valid or not. If valid then user is verified and method returns true
showUserInfo() : void	This method uses the result from verifyUser() method to redirect the user to the MainStartUI activity where they can further view their data. If the user is not verified then there should not be any redirection but an error message instead.

class SettingsUI	
This class shows the screen activity with the settings of the application. It enables the user to change the notification settings by using the given buttons	
Attributes	
-private Button notificationOnOff -private Button soundOnOff	
Methods	
setNotificationSound(boolean v) : void	This method is triggered upon button press and it's used to set the notification sounds on or off, depending on the boolean value of the argument.
setNotificationState(boolean v) : void	This method is triggered upon button press and it is used to activate or deactivate notifications on the mobile application.

class ListOfCoursesUI	
This class shows the screen activity with some of the user profile data adjacent together with the list of courses they give lecture at	
Attributes	
-private ArrayList<CourseData> listOfCourses	
Methods	
showCourseInfo(arg courseSelected) : void	This method is used to redirect the user to the specific course data view based on the user input. The user thus has to press the course button in order for this method to be used.

class CourseViewUI	
This class shows the screen activity with the previously selected course and some textual data regarding the course. The user may also view the pictures of the students that are detected from the face recognition process through a grid view. The methods of this class will also the user the possibility to manually mark students present in case the face recognition was not well trained enough.	
Attributes	
-private CourseData courseSelected	
Methods	
searchStudent (int id) : boolean	This method is used to search students from the student list and then filter the results according to the user input.
markPresent(StudentData selectedStudent) : boolean	This method is used to manually mark a student present in class by making a request towards the server.
calculatePercentage() : void	This method is used to calculate the percentage of students present and not and show this through a graphical representation.

class CourseData	
This class is used to create the structure of a course and store its data in an object format easy to be reused. The object is thereafter used by other classes as well.	
Attributes	
-private String courseName -private String courseCode -private ArrayList<StudentData> listOfStudents	
Methods	
insertStudent(StudentData selected) : void	This method is used to insert a new student into the listOfStudents of the course. It is to be used when the application receives the data from the server.
markStudentPresent(StudentData selected) : boolean	This method is used to set the **present** attribute of a particular student to **true.**

class StudentData	
This class is used to store the main data regarding a single student. It is therefore used in CourseData class as well.	
Attributes	
-private int studentID -private String studentFaceID -private boolean present	
Methods	
getStudentID() : int	Get method
setStudentID() : boolean	Set method
getStudentFaceID() : String	Get method
setStudentFaceID(): boolean	Set method

3.2 Face Recognition Package Interface

class FaceRecognition	
This class finds and identifies individuals in the class.	
Attributes	
private String dlibModel private String torchModel private int DefaultDims = 96 private dlib.fhog_object_detector faceDetector private openface.TorchNeuralNet faceRecognizer	
Methods	
public getAttendance() : void	This method saves the individuals detected in the class in a text file.
public reportResults(String filePath) : void	This method reports the results to the database based on the file generated by the method getAttendance().
private getFaces(int[][] image): dlib.rectangles	This method accepts as an input an image, whereby it detects all the faces and then returns the location of those faces as dlib.rectangles.
private compareFaces(int[] f1, int[] f2): double	This method takes the vector representation of two faces and returns the value in double of how closely related these two faces are.

Interface Camera	
This interface is for the implementation of the integration of the camera module in the system	
Attributes	
Methods	
public takePicture(): int[][]	This method takes a picture with the module and returns the resulting array of the picture in grayscale.

3.3 Database Package Interface

class Student	
This class is used to store the main data regarding a single student. It is therefore used in Course class as well.	
Attributes	
-private int id -private String name -private List<Courses> courses -private List<Classes> classes	
Methods	
login (String username, String pasw) : boolean	This method will enable the student to log in the system if the credentials are correct.
makeObjection(String course) : void	This method will enable the student to make an objection in case the system doesn't count the student as present. A confirmation from professor is needed to count the student as present.
getAttendances (String course) : int	This method will show the number of times the student has attended a course.

class Professor	
This class is used to store the main data regarding a single profesor. It is therefore used in Course class as well.	
Attributes	
-private int id -private String name -private List<Courses> courses -private List<Classes> classes	
Methods	
login (String username, String pasw) : boolean	This method will enable the professor to log in the system if the credentials are correct.
confirmObjection(String course) : void	This method will enable the professor to confirm an objection in case the system doesn't count the student as present.

class Classes	
This class is used to store the main data regarding a single class. It is therefore used in Student and Professor class as well.	
Attributes	
-private int id -private String name -private String location -private List<Courses> courses	
Methods	
getters() and setters()	These methods will be used for getting and setting data

class DatabaseManagement	
This class is used to connect to the database and to provide and send data	
Attributes	
private int id private int username private String password private String email private Connection con	
Methods	
public connect (Connection con) : boolean	This method is used to make the connection with the database in the server.
public checkPassword(String username, String pasw): boolean	This method check the credentials of the user and returns true or false depending on the correctness of the provided data
public authenticate(String username) : void	This method enables a user to log in the system or modify personal data in case of correct credentials
public getClasses(String username) : List	This method returns the classes that a professor or student attends.
public getCourses(String username) : List	This method returns the courses that a student or professor attends

3.4 Communications Package Interface

class Encryption	
This class will be used to encrypt and decrypt outgoing and incoming messages.	
Attributes	
private int keyValue	
Methods	
public encrypt(String message) : String	This method will encrypt the outgoing messages using the keyValue.
public decrypt(String message) : String	This method will decrypt incoming messages using the keyValue.

class RequestHandler	
This class handles all the requests within the system.	
Attributes	
private Encryption decryptor private Encryption encryptor private Database database	
Methods	
private parseMsg(String message): Message	This method will parse the message into the appropriate Message implementation.
private authMsg(Message message): boolean	This method authenticates the message send such that the system only handles valid requests.
public execMsg(String message): void	This method will be called when a message arrives. This then calls all the other private methods to handle and execute the request appropriately.
private sendReqDb(String request): void	This method sends a request to the database, such as saving data to the database
private sendNotAndroid(String request): void	This method sends notifications to the Android application.
private sendDataUnit(String data): void	The method is used to send data to the board unit, such as requested data from the database.

Inteface Message	
This interface is for the implementation of different message formats.	
Attributes	
private String type private String sender private String receiver	
Methods	
public getMessageType() : String	This method will return the type of the message.
public getSender() : String	This method will report the sender who sent the message.
public getReceiver() : String	This method will report the receiver of the message.