Application Design Document

Emotion Detection System for Students

Members

Ali Jbareen alijb@post.bgu.ac.il
Salih Qadri qadri@post.bgu.ac.il

Index

4 **Chapter 1 - Use Cases**
4 Use Case 1: User Registration
4 Use Case 2: User sign-in
5 Uses Case 3: Student Emotion
5 Uses Case 4: Student negative emotion
6 Uses Case 5: filtering for a specific set of emotions
6 Uses Case 6: choose a set of emotions to get notifications on
7 Uses Case 7: display statistics table for student emotions
7 Uses Case 8: user sends messages to other users in same room
8 Uses Case 9: Student Voice

9 **Chapter 2 - System Architecture**

10 **Chapter 3 - Data Model**

13 **Chapter 4 - Behavioral Analysis**
13 Sequence diagram 4.1
13 Uses Case 7: display statistics table for student emotions
13 Uses Case 3: Student Emotion

14 **Chapter 5 - Object-Oriented Analysis**
14 Class Diagrams 5.1
14 Class descriptions - 5.2
14 Agent:
14 Agent API - Flask:
16 CNC:
17 CNC API - Flask:
19 Packages - 5.3

20 **Chapter 6 - User Interface**
21 Log in page
21 Home page:
22 Malwares page:
23 Agents page:
23 Add new Malware:
24 Agents data:
25 System Log:

26 **Chapter 7 - Testing**

Chapter 1 - Use Cases

Use Case 1: User Registration

Primary Actor: student/teacher

Preconditions: None

Postconditions: the system save the user data

Basic Flow:

the user enter the URL link

the user click registration button

the system display a form to the user and ask it to fill it with his name + email + password + (teacher/student)

the student fill the form and click register button

the system saves the user data

Use Case 2: User sign-in

Primary Actor: student/teacher

Preconditions: the user registered to the system

Postconditions: The user is signed in

Basic Flow:

the user enter the URL link

the user click sign-in button

the system display a form to the user and ask it to fill it with his email + password

the user fill his email and password and click the sign-in button

if the credentials is correct: user will log in to the system

else : proper message will be displayed

Uses Case 3: Student Emotion

Primary Actor: student.

Preconditions:

student enter the teacher class/group

student's camera is opened

Postcondition: teacher dashboard changed to the current state of the student

Basic Flow:

the camera capture the student

the system detect his face in real time

the system detect his emotion from his face i real time

the system send to the teacher dashboard the frame and the face detection and emotion detection data

the dashboard change current student frame and draw a box on his face with his current emotion

Uses Case 4: Student negative emotion

Primary Actor: student

Preconditions:

student enter the teacher class/group

student's camera is opened

Postcondition: none

Basic Flow:

(UserCase 3)

then the system check recognize that the student has a negative emotion

the dashboard notifies the teacher with a relevant notification.

Uses Case 5: filtering for a specific set of emotions

Primary Actor: teacher

Preconditions:

student enter the teacher class/group

student's camera is opened

teacher enter the system

Postcondition: none

Basic Flow:

the teacher select a set of emotions on his dashboard

the system filter the all students' emotions and display only the set that the teacher choose

Uses Case 6: choose a set of emotions to get notifications on

Primary Actor: teacher

Preconditions:

student enter the teacher class/group

student's camera is opened

teacher enter the system

Postcondition: none

Basic Flow:

the teacher select a set of emotions on his dashboard that he what the system to notify him on

the system disable "Student negative emotion" notification

the system notify the teacher when a student change his emotion to these emotions

Uses Case 7: display statistics table for student emotions

Primary Actor: teacher

Preconditions:

student enter the teacher class/group

student's camera is opened

teacher enter the system

Postcondition: none

Basic Flow:

the teacher Click on a statistic button

the system display on the dashboard a statistic table with all students emotions during the activity

Uses Case 8: user sends messages to other users in same room

Primary Actor: suser

Preconditions: user entered the teacher class/group

Postcondition: none

Basic Flow:

Students type a message in a chat form and press enter

the system sends the message to the group online members

the system display the message with his user name in the chat form

Uses Case 9: Student Voice

Primary Actor: student

Preconditions:

student is logged in

Postcondition: none

Basic Flow:

the system student click on enable mic button

the student start to talk

the system capture his voice in real time

the system send his voice to the dashboard in real time

the dashboard plays the student voice in real time

Chapter 2 - System Architecture

The system is comprised of 3 servers:

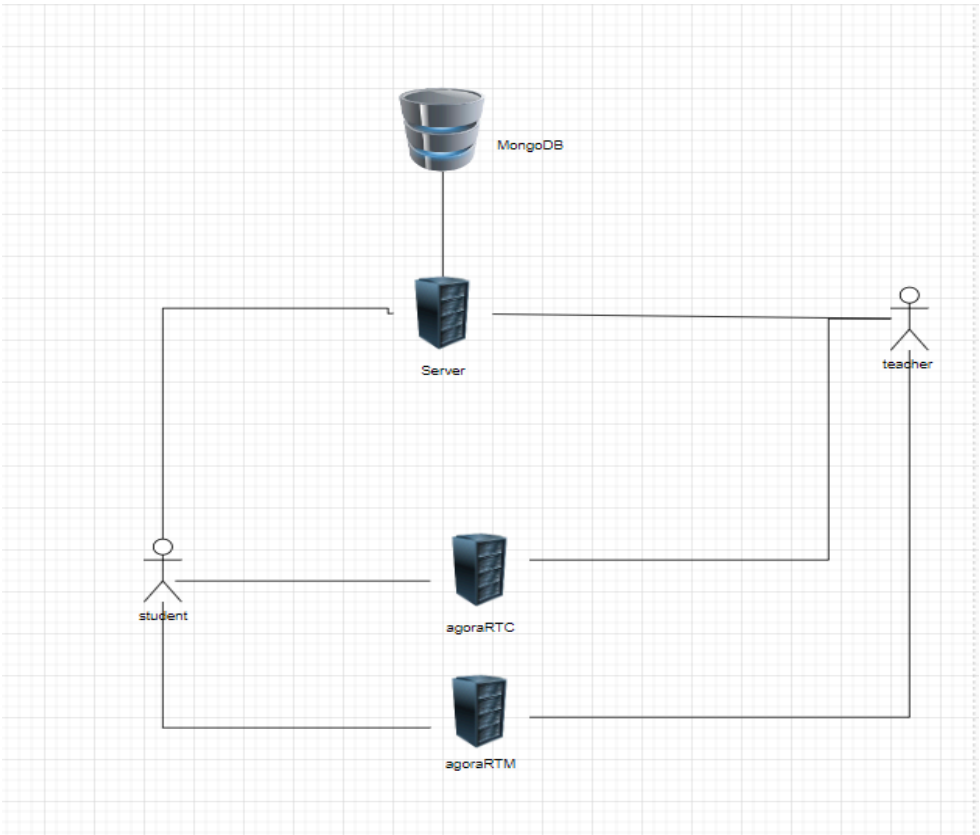
1) Client:- this component is responsible for capturing the student face and processing it and send the result to the server to save them and to the teacher Dashboard to display them in real time

2) Server (NodeJS) deployed in Heroku

Server deployed in Heroku cloud, using express framework to handle user requests, authenticate user requests, communicate with the database, save user emotions statistics.

3) RTC Server- used as TURN server to relay video and audio traffic from student browser to the teacher browser

4) RTM Server- used as TURN server to relay regular messages and emotion messages from student browser to the teacher browser



Chapter 3 - Data Model

3.1 Description of Data Objects

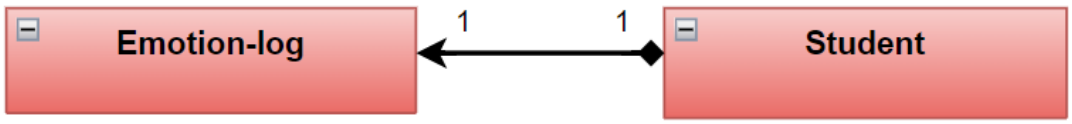
Student : each student must have a unique id , name ,gender ,age ,and what Class-Room , and his teacher name , also ,each student has a log of his emotion during the class for the Analysis.

Teacher : each teacher have unique id and Name, And Class-Rooms that he can view their cams , and for each student that participate in his room , teacher can view statistics of the room .

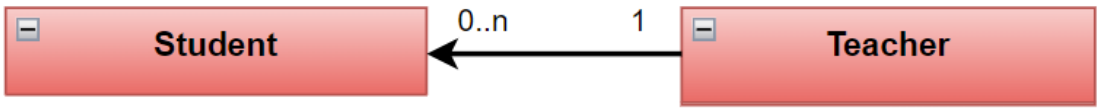
Emotion-log : has a student's unique ID , each student have his own emotion log during all the classes he participated in .

Class-Room : each room have a unique id , and only students who are permitted to enter that room can enter

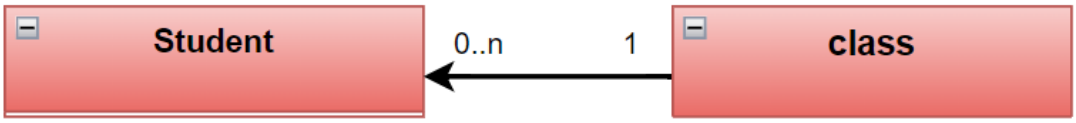
3.2 Data Object Relationships



each student has his own emotion log , captured during the class time for further statistics

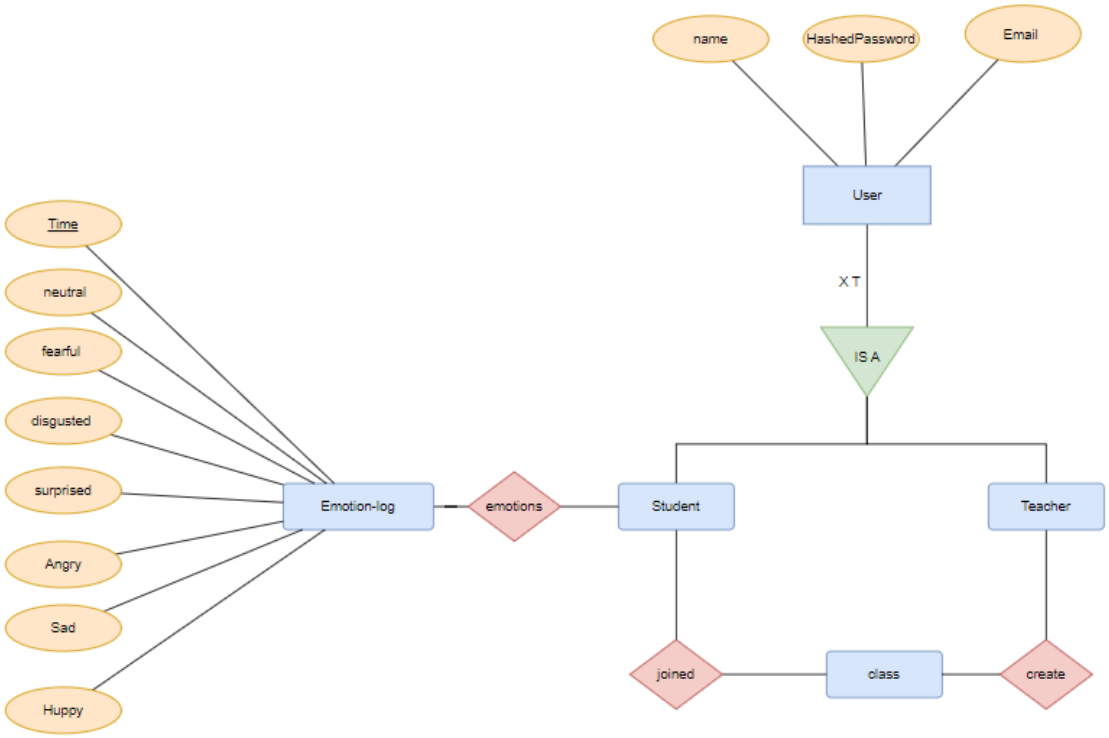


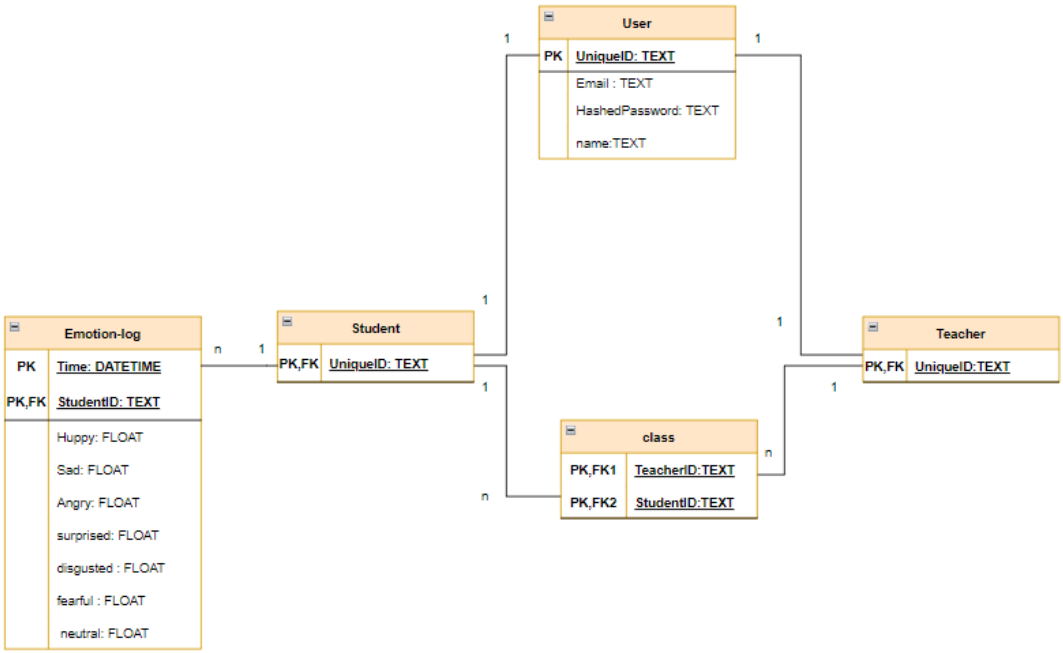
Every student has one teacher that can Access his camera and logs



each student has one room that can Enter

3.3 Databases

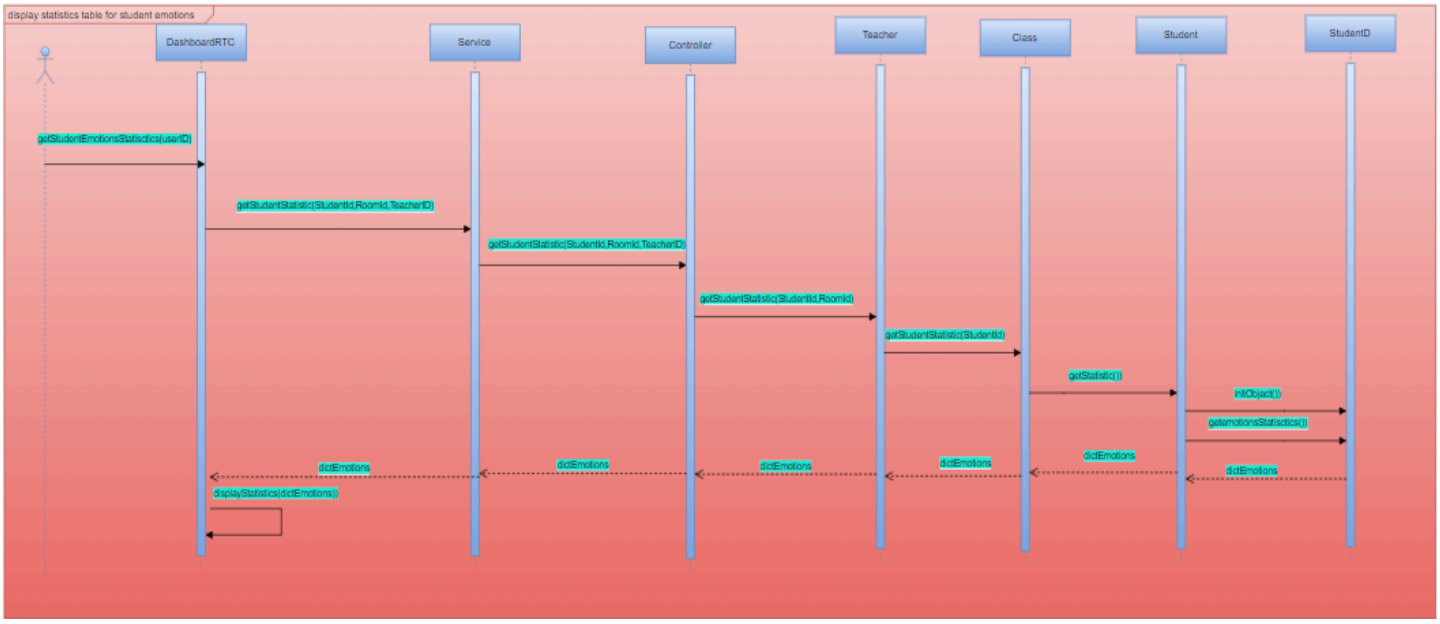



User
PK
UniqueID: TEXT
Email : TEXT
HashedPassword: TEXT
name:TEXT
Emotion-log
PK
Time: DATETIME
PK,FK
StudentID: TEXT
Huppy: FLOAT
Sad: FLOAT
Angry: FLOAT
surprised: FLOAT
disgusted : FLOAT
fearful : FLOAT
neutral: FLOAT
Student
PK,FK
UniqueID: TEXT
Teacher
PK,FK
UniqueID:TEXT
class
PK,FK1
TeacherID:TEXT
PK,FK2
StudentID:TEXT
1
n
1
1
1
1
1
1
n
n

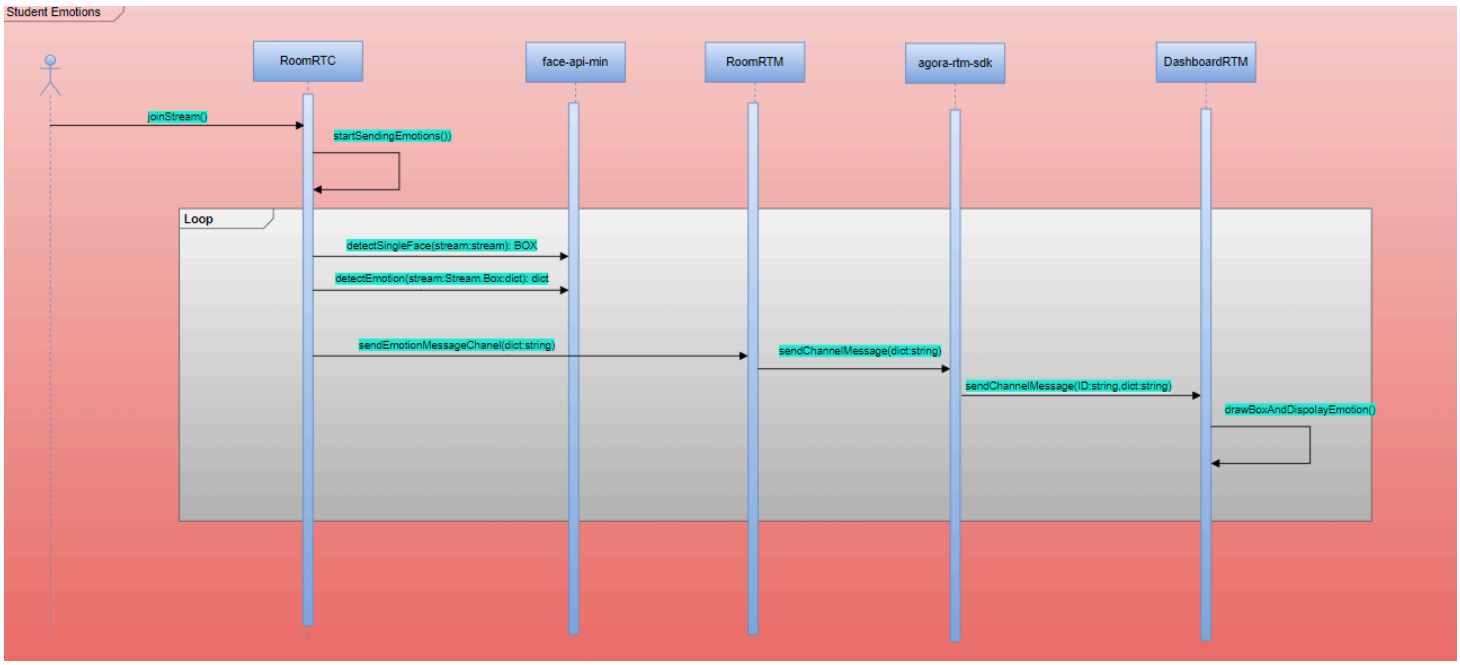
Chapter 4 - Behavioral Analysis

4.1 Sequence diagram

Uses Case 7: display statistics table for student emotions

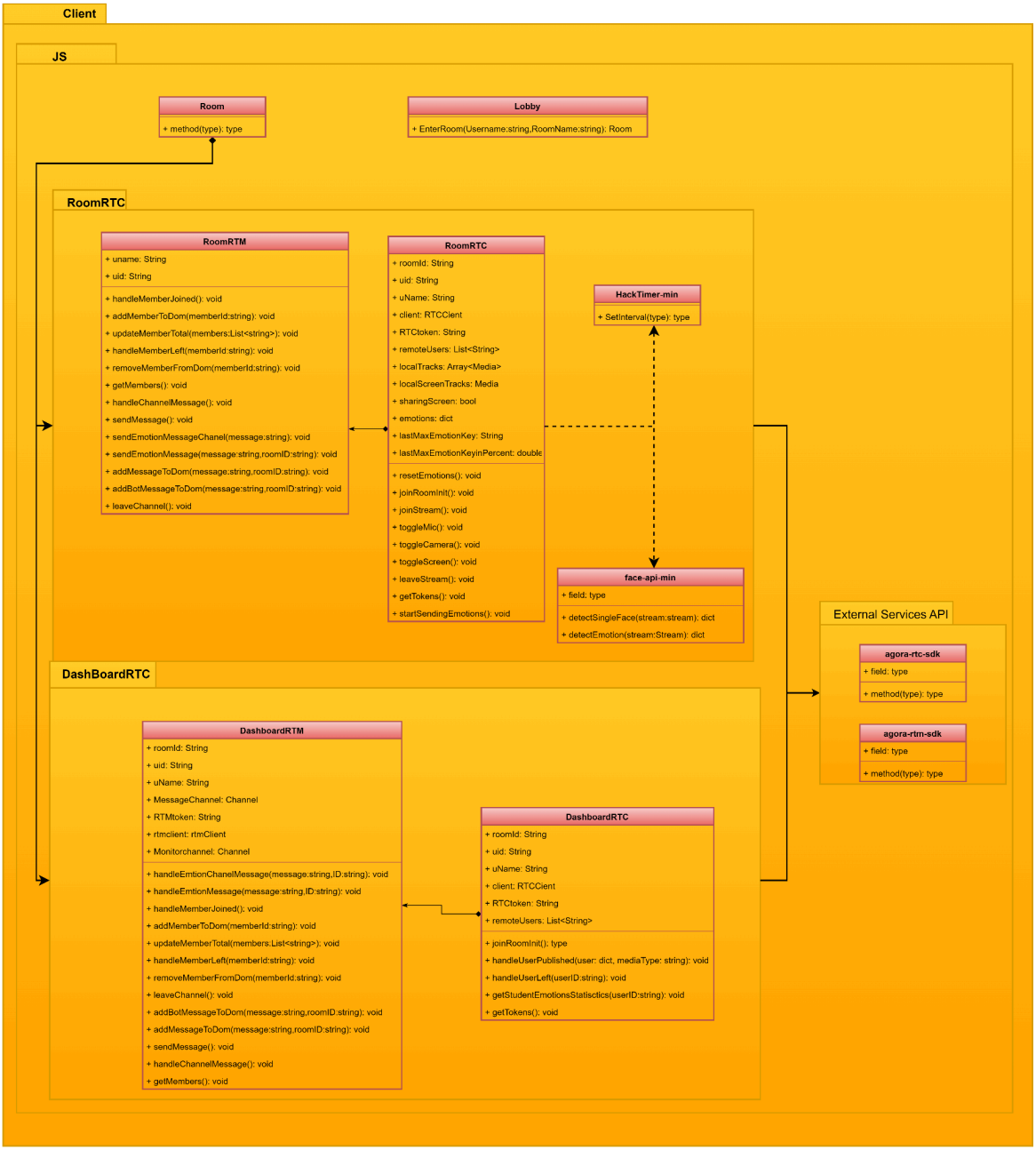


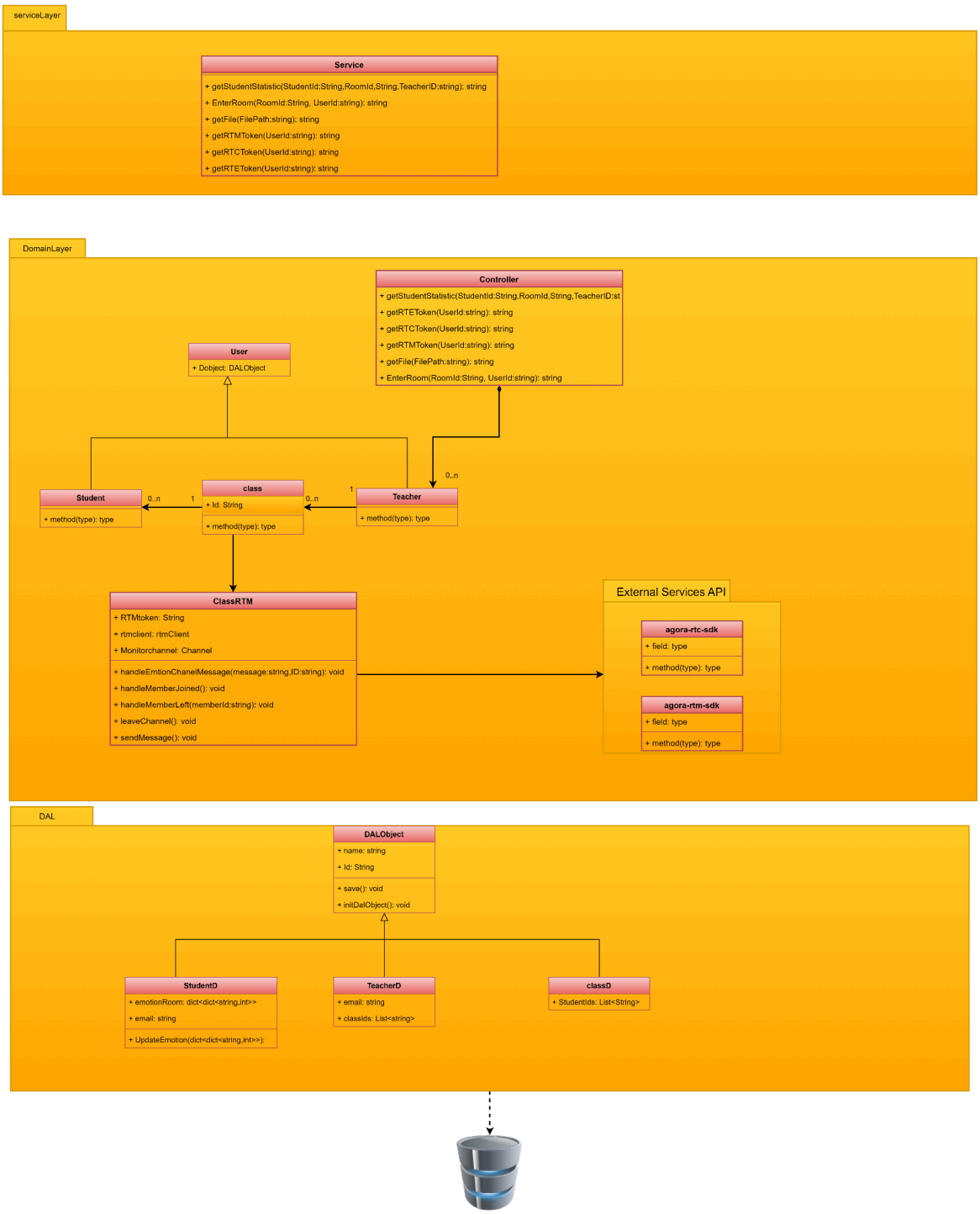
Uses Case 3: Student Emotion



Chapter 5 - Object-Oriented Analysis

5.1 Class Diagrams + 5.3 Packages




serviceLayer
Service
+ getStudentStatistic(StudentId:String,RoomId,String,TeacherID:string): string
+ EnterRoom(RoomId:String, UserId:string): string
+ getFile(FilePath:string): string
+ getRTMToken(UserId:string): string
+ getRTCToken(UserId:string): string
+ getRTEToken(UserId:string): string
DomainLayer
Controller
+ getStudentStatistic(StudentId:String,RoomId,String,TeacherID:st
+ getRTEToken(UserId:string): string
+ getRTCToken(UserId:string): string
+ getRTMToken(UserId:string): string
+ getFile(FilePath:string): string
+ EnterRoom(RoomId:String, UserId:string): string
User
+ Dobject: DALObject
Student
+ method(type): type
0..n
1
class
+ Id: String
+ method(type): type
0..n
1
Teacher
+ method(type): type
0..n
ClassRTM
+ RTMtoken: String
+ rtmclient: rtmClient
+ Monitorchannel: Channel
+ handleEmtionChanelMessage(message:string,ID:string): void
+ handleMemberJoined(): void
+ handleMemberLeft(memberId:string): void
+ leaveChannel(): void
+ sendMessage(): void
External Services API
agora-rtc-sdk
+ field: type
+ method(type): type
agora-rtm-sdk
+ field: type
+ method(type): type
DAL
DALObject
+ name: string
+ Id: String
+ save(): void
+ initDalObject(): void
StudentD
+ emotionRoom: dict<dict<string,int>>
+ email: string
+ UpdateEmotion(dict<dict<string,int>>):
TeacherD
+ email: string
+ classIds: List<string>
classD
+ StudentIds: List<String>

5.2 - Class descriptions

RoomRTC: responsible for voice and audio communication ,facade of the RoomRTC package.
invariants:

- roomId: unique constant and not empty
- uid: unique constant not empty
- uName: user name, not empty
- lastMaxEmotionKey natural number from 0 to 6
- lastMaxEmotionKeyinPercent real number from 0 to 100

methods:

- joinRoomInit()
  pre condition:
  1. valid RTM and RTC tokens
  2. not called before

  post condition:
  1. client not null
  2. localTracks is array of size 2 with not null values
  3. sharingScreen is false
  4. RoomRTM initialized with uid and UName
- joinStream()
  pre condition:
  1. joinRoomInit method called before
  2. valid RTM and RTC tokens
- startSendingEmotions()
  pre condition:
  1. valid RTM and RTC tokens
  2. student joined the stream by calling joinStream()

RoomRTM: responsible for real time messaging.
methods:

- sendEmotionMessageChanel(message:string)
  pre condition:
  1. valid RTM tokens
  2. message is json format

  post condition:
  1. DashboardRTM::handleEmtionMessage(message:string,ID:string) triggered with same message and student ID

DashboardRTC: responsible for voice and audio communication ,facade of the dashboardboard package.
invariants:

- roomId: unique constant and not empty
- uid: unique constant not empty
- uName: user name, not empty

methods:

- joinRoomInit()
  pre condition:
  3. valid RTM and RTC tokens
  4. not called before

  post condition:
  5. client not null
  6. DashboardRTM initialized with uid and UName
- handleUserPublished(user: dict, mediaType: string): void
  pre condition:
  1. media type is “video” or “audio”

2. user contains the user id

- getStudentEmotionsStatisctics(userID:string)
  pre condition:
    valid RTM and RTC tokens
    student with userID joined this room before

DashboardRTM: responsible for real time messaging.
methods:

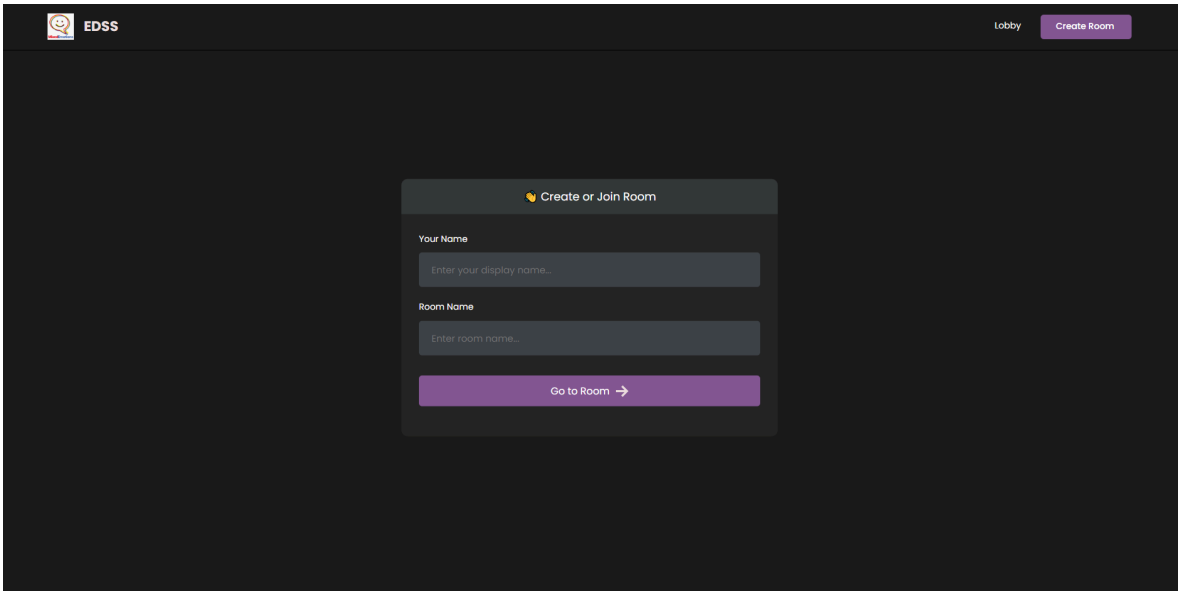
- handleEmtionChanelMessage(message:string,ID:string)
  pre condition:
  1. valid RTM tokens
  2. message is json format

5.4 Unit Testing:

1. invariants: call every method in roomRTC and in roomRTM once and check the invaiants between calls
2. pre conditions:call every method with invalid precondition and check if it's throwing exception
3. post conditions: call every method with valid precondition and check if it's not throwing exception and postconditions are met

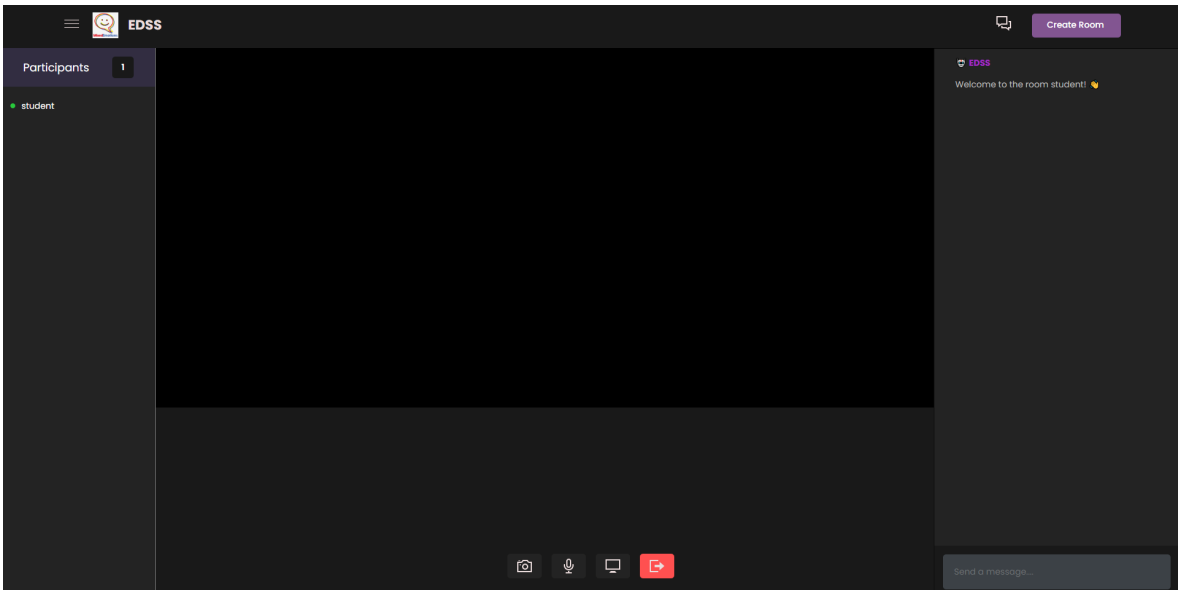
Chapter 6 - User Interface

Lobby:



description:before entering a room user will enter his name and the roomName, if roomName is empty the system will generate a unique room

Student Room:



description:after entering the room this page will be loaded were the student video will be displayed on the page and will get these buttons under his video where he can open/close

camera ,open/close microphone ,share his screen,leave room,sent messages to the teacher and see all participants (users who joined the room)

Teacher Dashboard:

description:after entering the room this page will be loaded where the students video will be displayed on the screen and their emotions will be added on there videos and can be expanded by clicking on there video or there names in the participants dev and to get the emotions statistics.

Chapter 7 - Testing

In order to test our system we will do several experiments where at least 15 students join the system and started solving geometric problems and at least one teacher joins the system to monitor the students' emotions, we will record the experiments and get feedback from the students and the teachers about the quality, usability and responsiveness of the system.