# Software Engineering

# Project's Documentation

# Student Faculty Document Sharing Android Project


**Team members:**

1. **Andrew Emil (20)**
2. **Bishoy Eshak (23)**
3. **Doaa Mohamed (27)**
4. **Mirona Gamil (73)**

# Requirements:

## ❖ User requirements:

Android application that shall allow instructors to upload documents and resources periodically from PC and allow students to have their access on these documents and download them on their smart phones or tablets.

## ❖ System requirements:

- Users login by their emails and password, emails are unique.
- Users should have their own accounts and system stores them in its database.
- User's account must contain information about his department, type (student or instructor) and semester.
- Administrator's email and password are encrypted in file.
- Administrator should assign courses to each student according to their semester and if they have courses from previous semesters.
- Administrator access and change all database's tables.
- Instructor database should also specify courses they are responsible of and they are allowed to upload files to their courses only.
- Instructors can't download any file.
- System send notifications to students when a new file is uploaded to a course they are enrolled in.
- Departments are: computer and systems engineering, electrical power engineering, communication and electronics engineering, mechanical engineering.
- Each department divided into 8 semesters.
- The home page of account contains courses that student or instructor enrolled in it and general file of the semester.
- Student's home page contains also total number of notification that he doesn't read.
- Categories of semester's files are: grades, timetable.
- Categories of course's files are: lecture notes, sheets, references, assignments, grades, instructors' office hours or any announcement by the instructors.
- Course's home page contains file uploaded in it and its notifications.
- Grades contain: sheet grades, assignment grades, midterm grades and final grades.

## ❖ Validation:

- Students can’t upload files.
- Instructors can’t download files.
- Students can only access courses of his semester and only courses that he has from previous semesters.
- Instructors can only access courses that they are teaching.

## ❖ Non-functional requirements:

- Email is unique.
- Course contents are available for all students enrolled in the course.
- Uploaded file size can't exceed 50 MB.

## ❖ Scenarios:

**First scenario:**

Initial assumption: student wants to check his account for new documents.

Normal: student logs in to the systems by providing his name and password, chooses notified course, check for new documents, announcements, ..etc and download desired file.

What can go wrong:

- Student may type wrong password or email, which should be handled by error message “wrong email or password”.
- Instructor may remove file before the student see it, which should be handled by message “file is no longer available”.
- When student check for new updates, he/she can't find new files.

**Second scenario:**

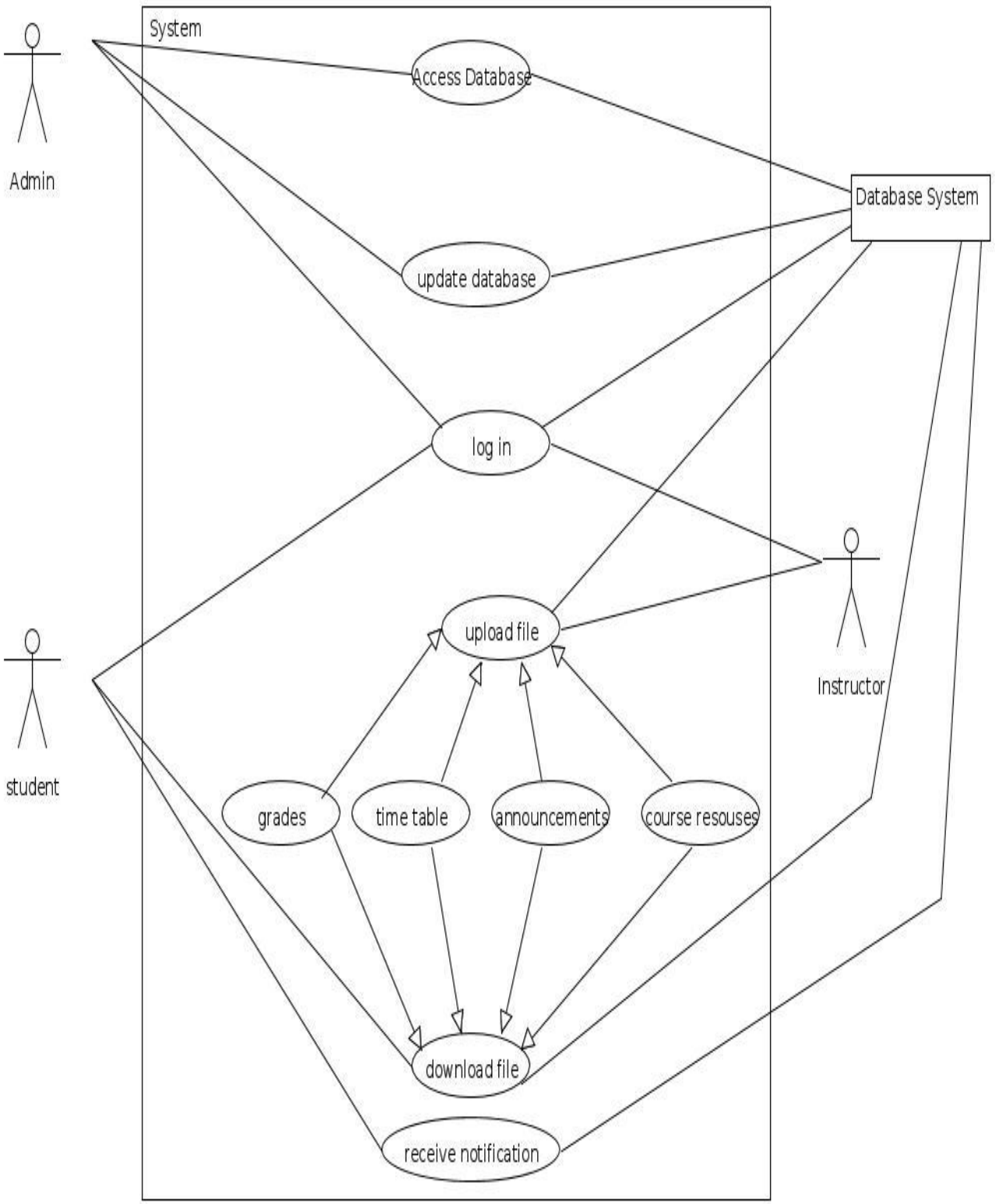
Initial assumption: instructor would like to upload new file to course page of his.

Normal: instructor logs in to the systems by providing his name and password, chooses the course, and upload the file.

What can go wrong:

- Instructor may type wrong password or email, which should be handled by error message “wrong email or password”.
- Instructor may upload file that exceeds 50 MB that should be handled by message “file size is too large”.

## Use case diagram:



# Tabular of each use case:

- **Use case log in:**

| System faculty file sharing | |
|---|---|
| Use case | log in |
| Actors | Admin, student, instructor, database system |
| Description | The program takes the email and password of the user and compares it to the admin's data, then to instructor's data and student's data. It checks whether the logged in user is the admin, a student, an instructor or invalid data. If invalid user re-enters the email and password. |
| Stimulus | User enters email and password and presses log in |
| Response | It checks whether the logged in user is the admin, a student, an instructor or invalid data. If it is valid the program saves the user data to use it latter on. If invalid user re-enters the email and password. |
| comments | User must log in to use any of the functionality of the program. |

- **Use case access data:**

| System faculty file sharing | |
|---|---|
| Use case | access data |
| Actors | Admin, system database |
| Description | The admin after logging in can access database. The admin can view any of its tables by choosing access database and requesting table name. |
| Stimulus | Choosing access data and providing the table name and pressing show table. |
| Response | Shows the table on screen. |
| comments | This use case is enabled if user logs in and he is the admin. |

❖ **Use case update data:**

| System faculty file sharing | |
|---|---|
| Use case | update data |
| Actors | Admin, system database |
| Description | The admin after logging in can update database. The admin can update any of its tables by selecting update.Then admin chooses table and kind of update he wants ( update – delete – insert ) then enters criteria for update or delete and enters data for update or insert. * Delete condition cannot be empty. |
| Stimulus | Choosing update data and providing data for update then pressing execute button |
| Response | Update the data base. |
| comments | This use case is enabled if user logs in and he is the admin. |

❖ **Use case upload file:**

| System faculty file sharing | |
|---|---|
| Use case | upload file |
| Actors | Instructor, database system |
| Description | The instructor after logging in can upload files to only the courses he teaches. So he chooses the course then the category of the file he is going to upload whether it is ( lecture notes – sheet – assignment – etc ). Then the user chooses a file for uploading then clicks upload. So file is uploaded and added to database and also notification for it is added to students enrolled in the course. |
| Stimulus | Instructor chooses uploading a file. |
| Response | A file is uploaded to database and a notification is sent to students enrolled in this course. |
| comments | This use case is enabled if user logs in and he is an instructor. |

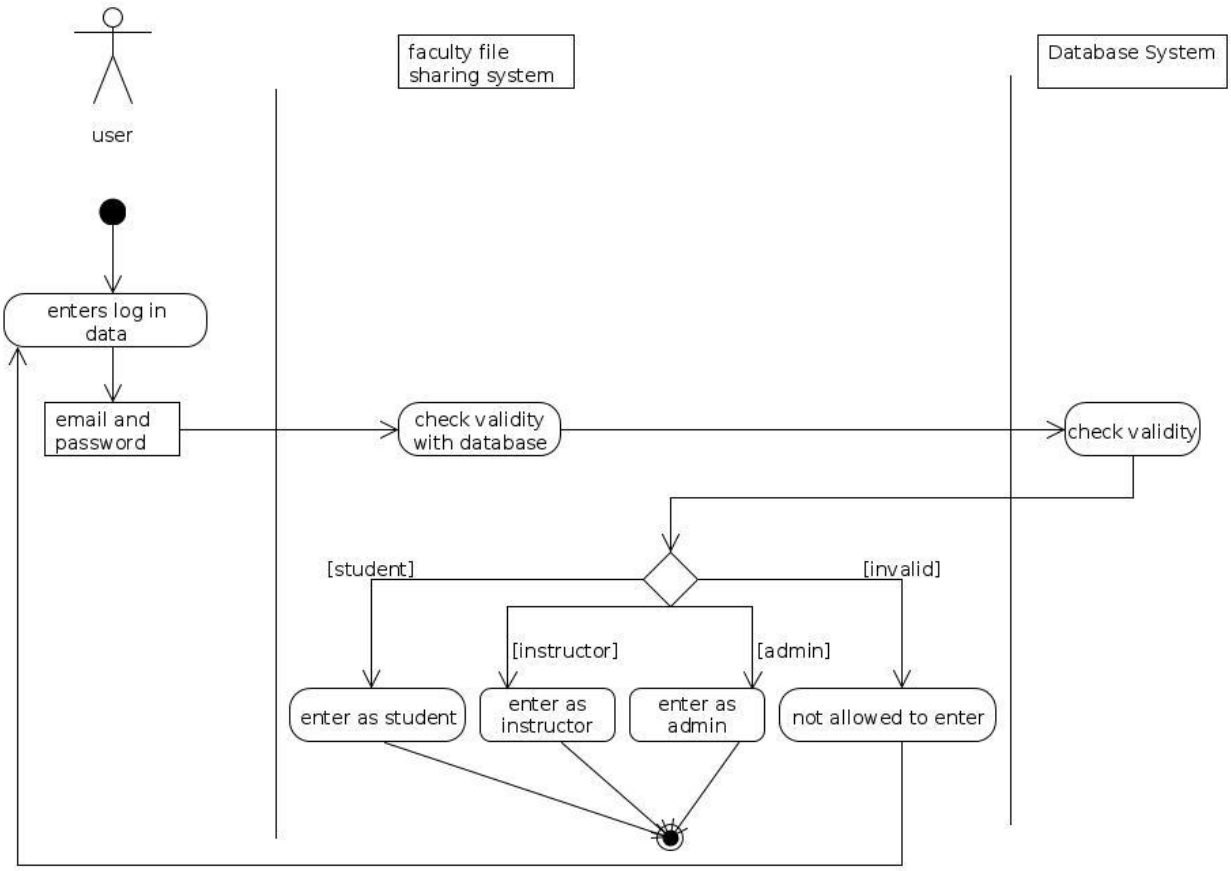
❖ **Use case download file:**

| System faculty file sharing | |
|---|---|
| Use case | download file |
| Actors | Student, database system |
| Description | The student after logging in can choose to download a file from semester contents in the semester he is registered in and also any special courses he has. The student chooses one of semester categories ( courses – grades – time table ). Then in courses it shows the categories of course contents ( lecture notes – sheet – assignment – etc ) and in each category shows the files available for download. By pressing the download button the file will be downloaded from database. |
| Stimulus | Student opens semester content and chooses a file to download and presses download button. |
| Response | Downloads file from server. |
| comments | This use case is enabled if user logs in and he is a student. |

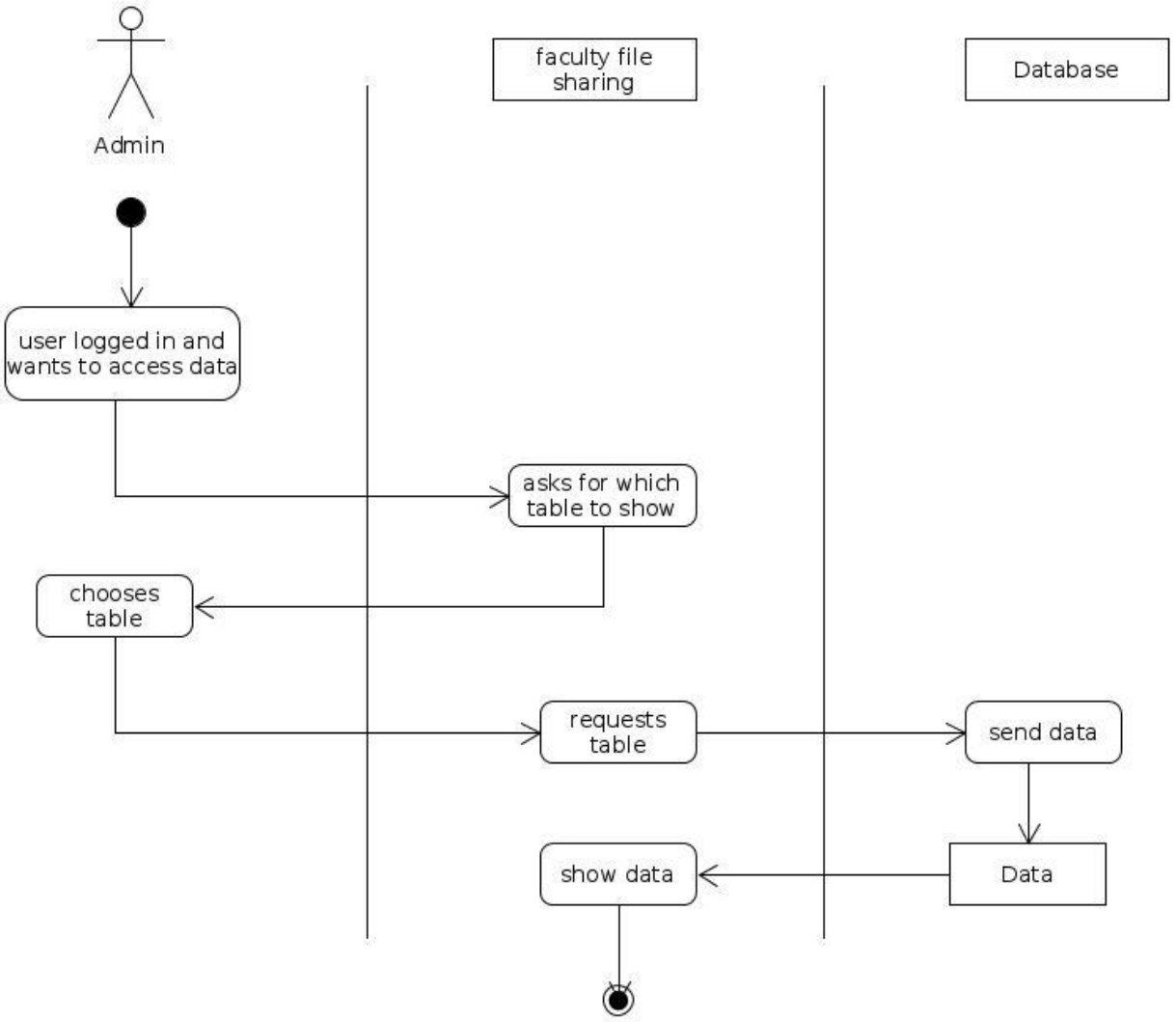
❖ **Use case receive notification:**

| System faculty file sharing | |
|---|---|
| Use case | receive notification |
| Actors | Student, database system |
| Description | The student after logging in can choose notification option. By choosing it it downloads notifications from server and when student clicks on notification it is automatically deleted from database. |
| Stimulus | Student opens semester content and chooses notifications. |
| Response | Get notifications from database and shows it on screen and when notification pressed it is deleted from database. |
| comments | This use case is enabled if user logs in and he is a student. |

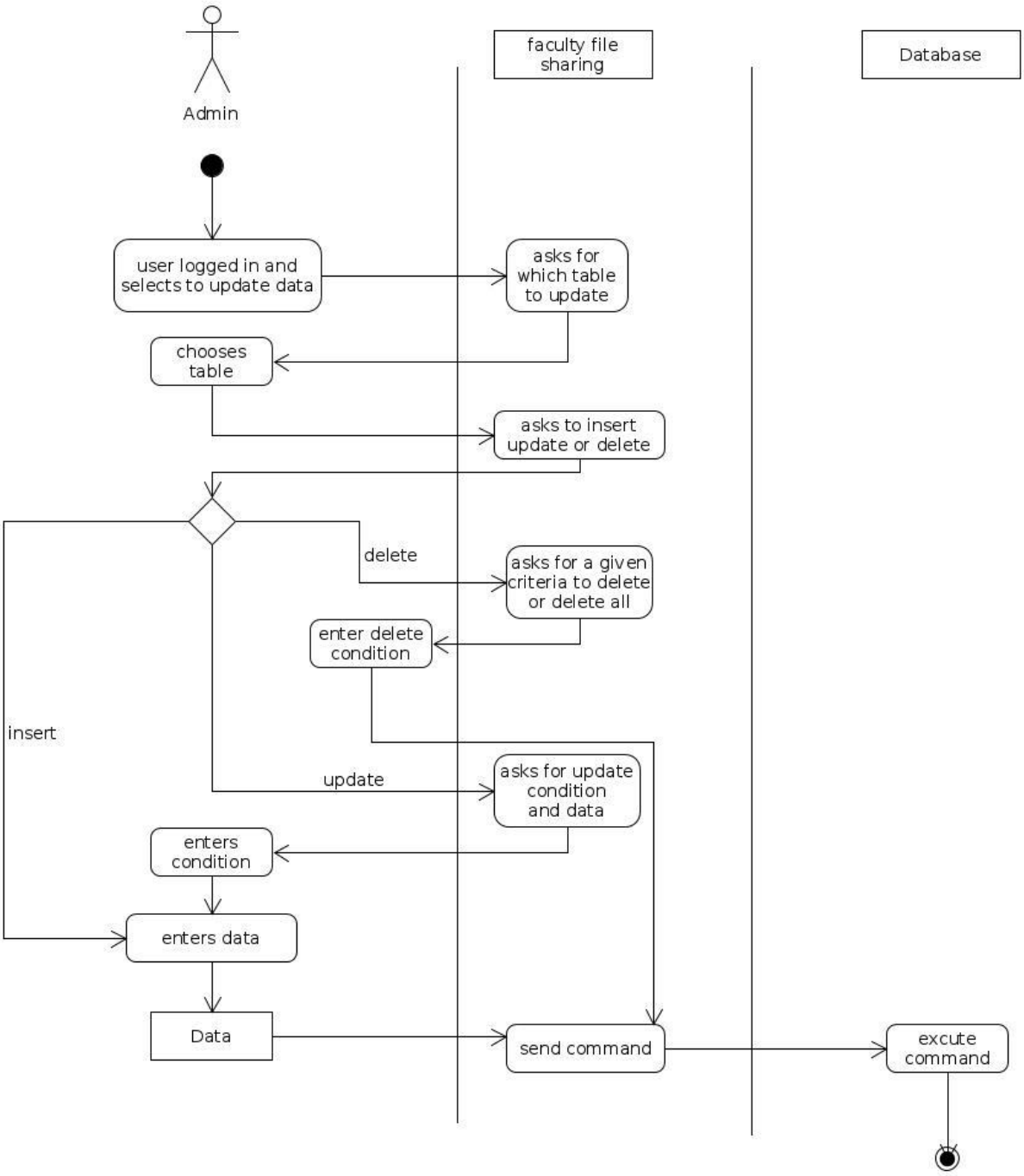
## Flow of events for each of the use cases using activity diagrams:




3.1 activity diagram of login




3.2 activity diagram of access data



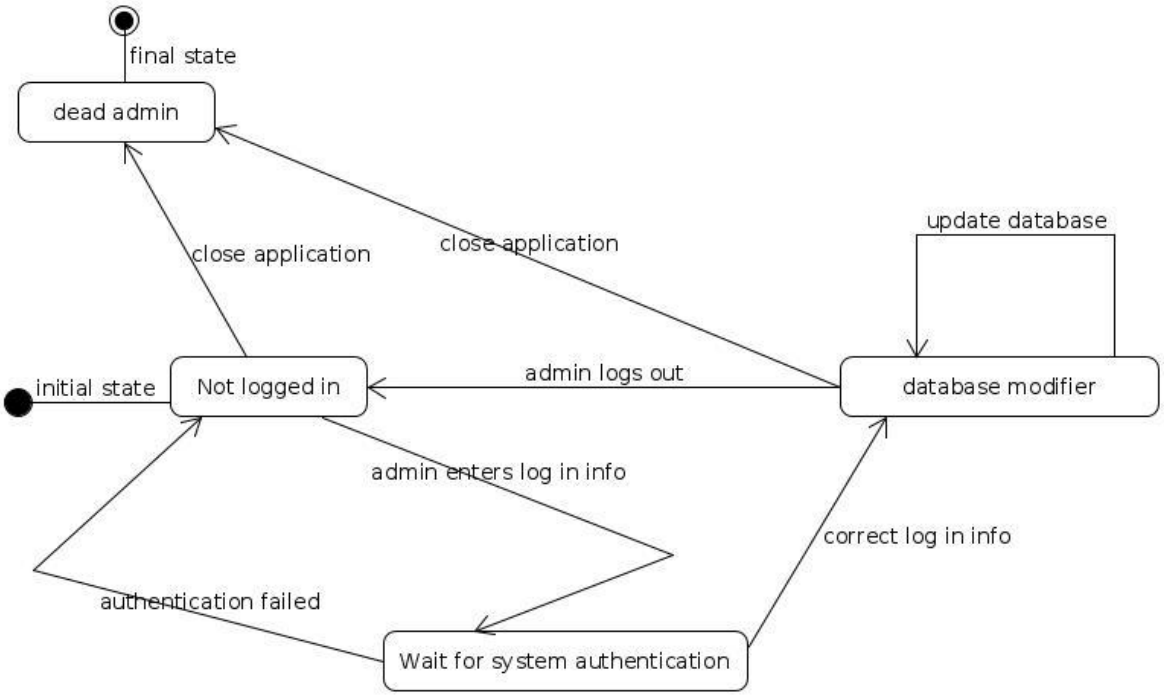

3.3 activity diagram of update data

## CRC Cards:

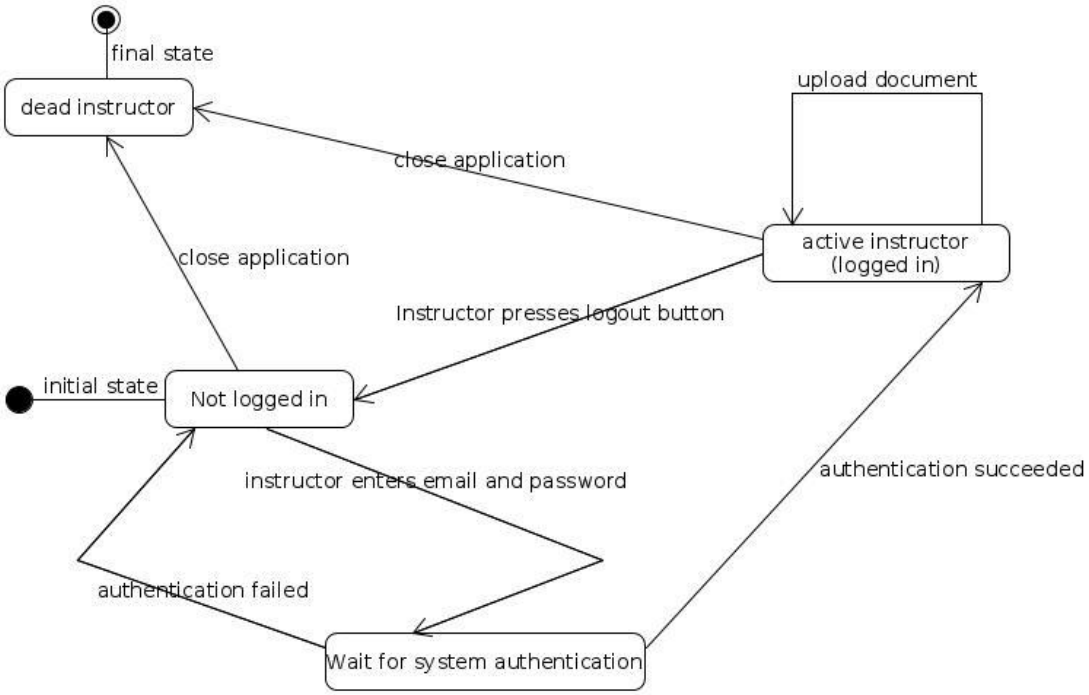
| class | responsibilities | collaborator |
|---|---|---|
| User | Contains ID, name, email and password of any user of the system | Admin<br>Instructor<br>Student |
| Admin | Access database<br>Change database | Account |
| Instructor | Can update file | Account |
| Student | Department and semester of student | Account |
| login | Takes email and password of user and validate it | User |
| Account | Show account of user according to its type (admin, instructor, student)<br>If admin show: update or change database.<br>If instructors show: his courses and option to download file to course<br>If student shows: courses of his semester and other courses | Notification<br>Database |
| Course_viewer | Display files uploaded in this course and show options to download if he is student | File_system<br>Database |
| Notifications | Send notification to students if file uploaded in his courses<br>Or receive notification that remove read notifications from student_notification | Account<br>Database |
| File_system | Gets file wanted to be downloaded from database of upload file to database | Notifications<br>Database |

| | | |
|---|---|---|
| Database | Access all tables of database | Instructor_data<br>Strudent_data<br>Cources<br>Special_courses<br>Files<br>Course_file<br>Genral_file<br>Notifications<br>Student_notification |
| Instructor_data | Data class contains id, name, email and password of each instructor | |
| Strudent_data | Data class contains id, name, email, password, department and semester of each student | |
| Cources | Data class contains id, name, instructor_id, semester and department of each course | |
| Special_courses | If student has courses out of his semester, this table contains student's id and course's id | |
| Files | Contains id, name and path of files | |
| Course_file | Class data contains course_id and category of files of courses | Files |
| Genral_file | Class data contains semester_id and category of general files of semester | Files |
| Notifications | Class data contains id, file_name, course_id and category | |
| Student_notification | Contains student's id and notification's id of unread notification | |

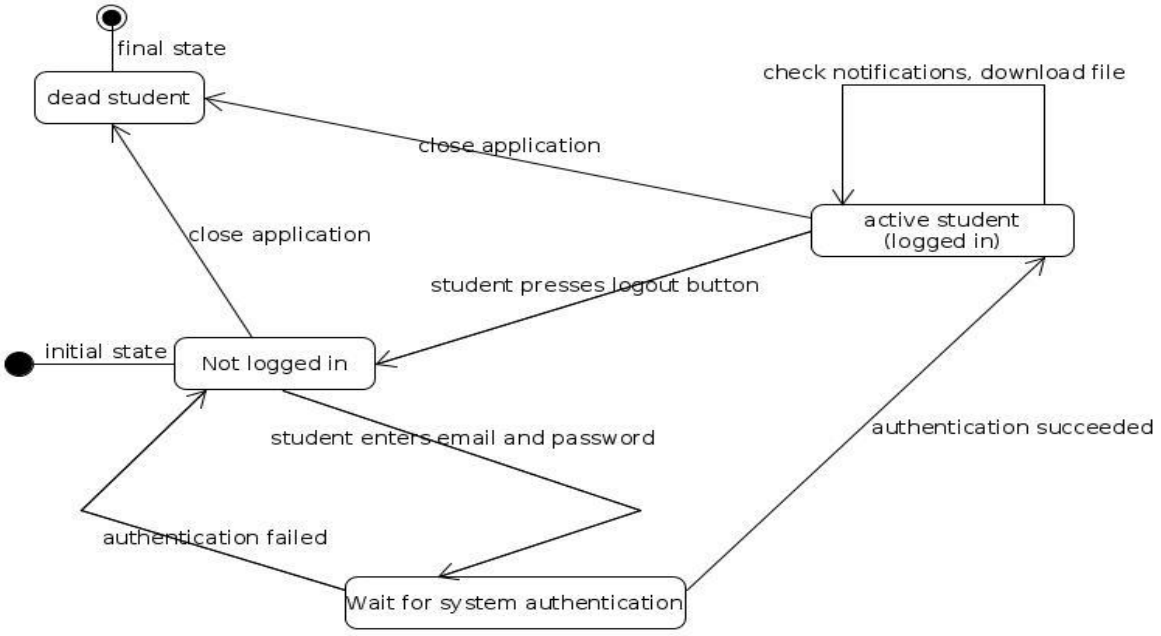
# Statechart diagrams:




5.1 statechart of admin




5.2 statechart of instructor



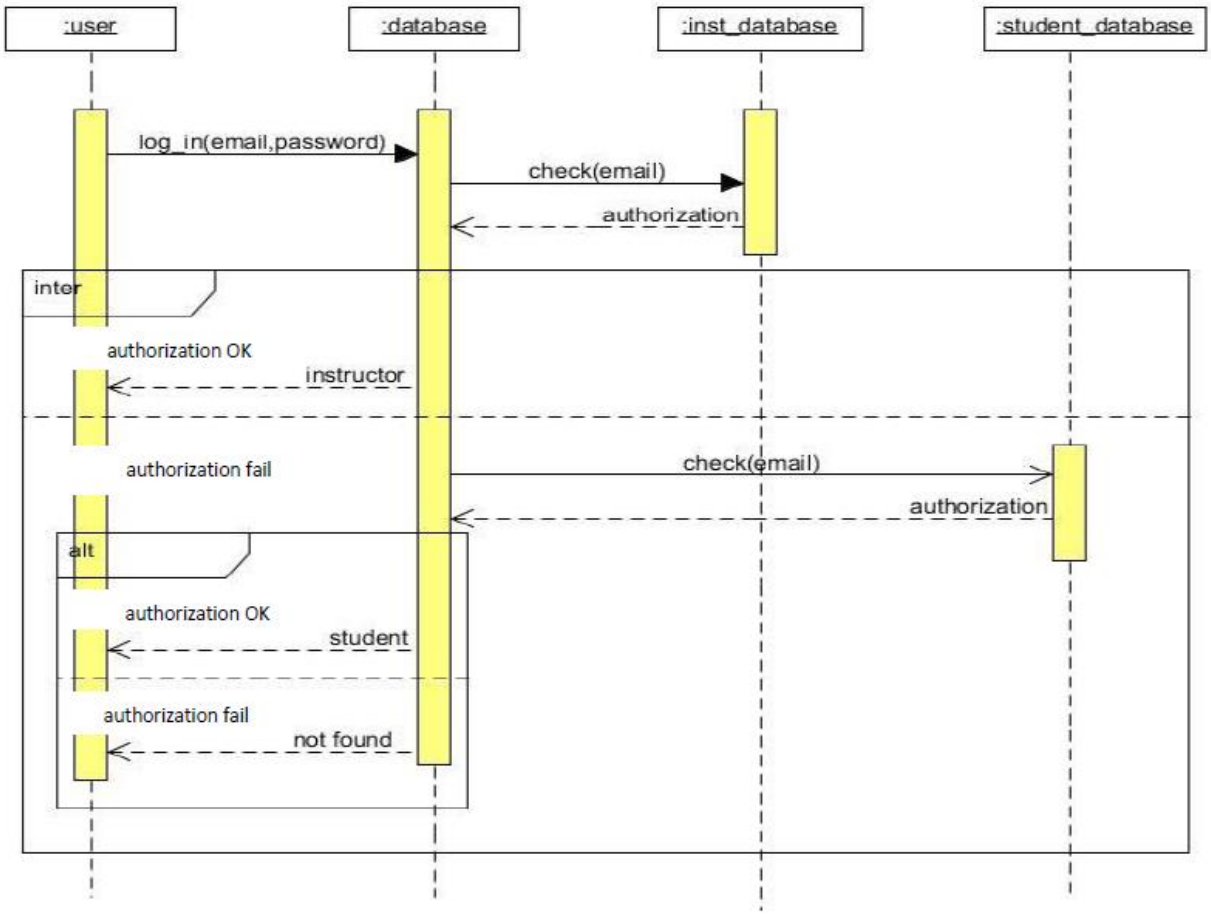

5.3 statechart of student

## Tabular of statechart:

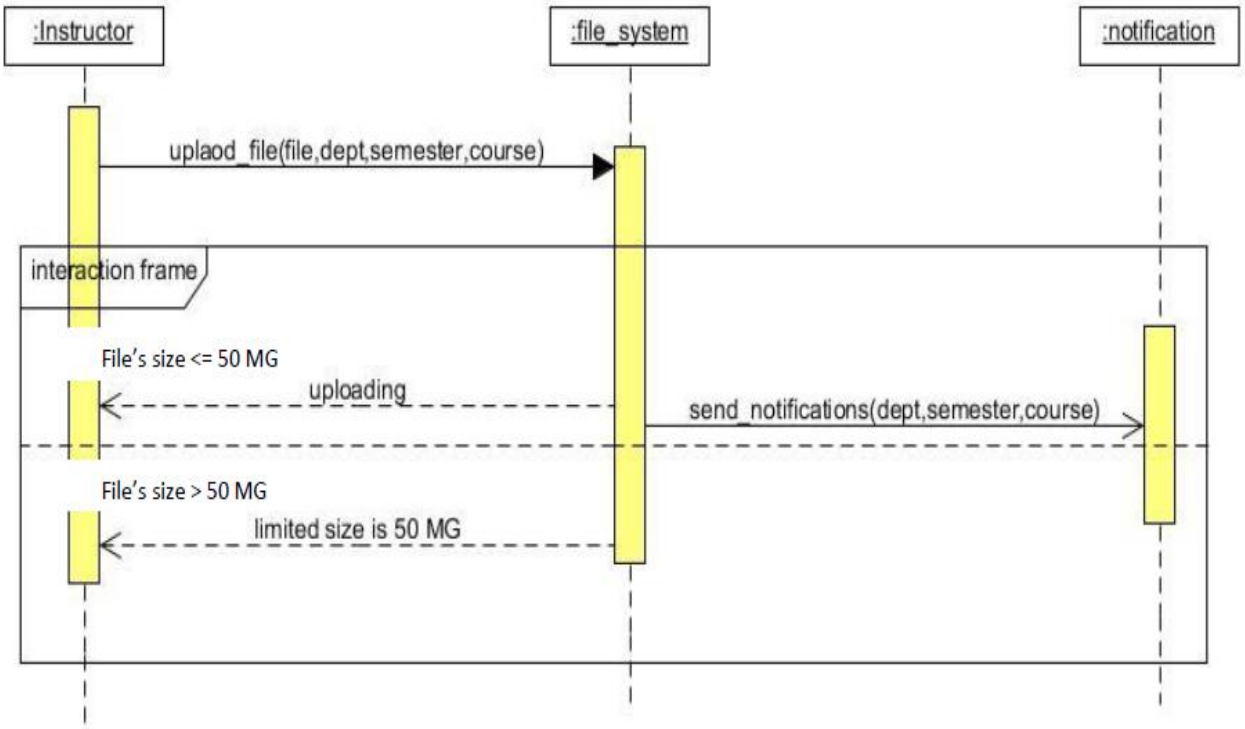
| State | Description |
|---|---|
| Not logged in | Represents user in login page and still didn't enter Email and password |
| Waiting for system authentication | Represents user waiting for system to validate his email and password |
| Active user | Represents user logged in successfully to his/her suitable account page and can use functions allowed to him/her as follows: if the user is:<br>-Student: he/she can check notifications and download files.<br>-Instructor: he/she can upload files.<br>-Administrator: he/she can modify database. |
| Dead user | Represent user when he/she close the application and is no longer using its functions |

# Interaction diagrams:

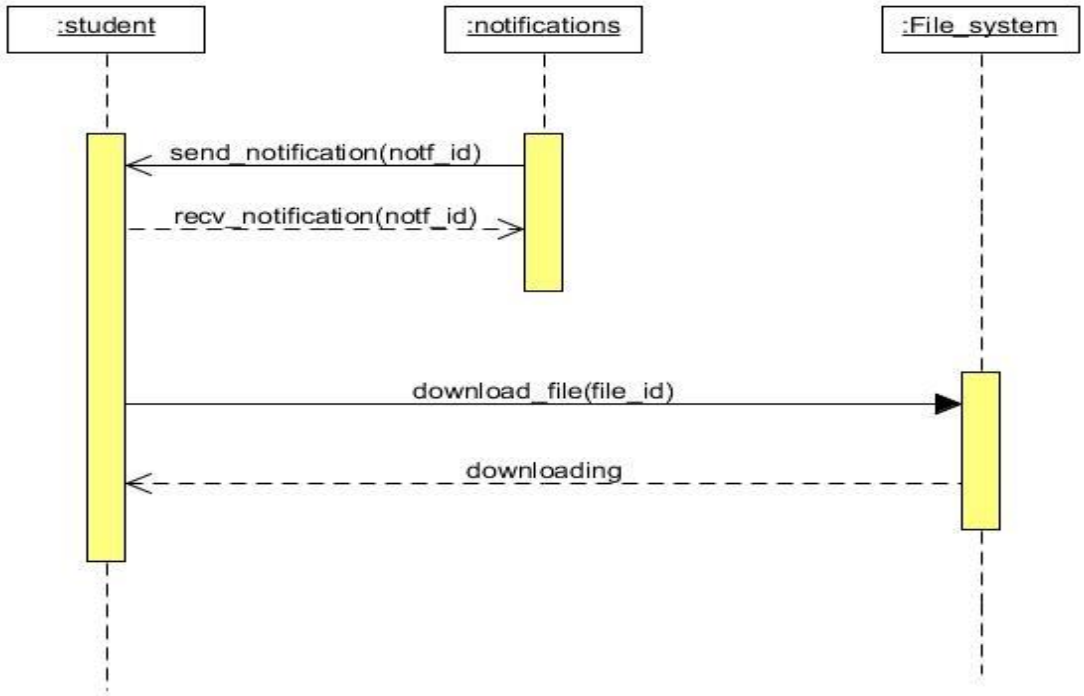
## ❖ Sequence diagram:




6.1.1 login sequence diagram

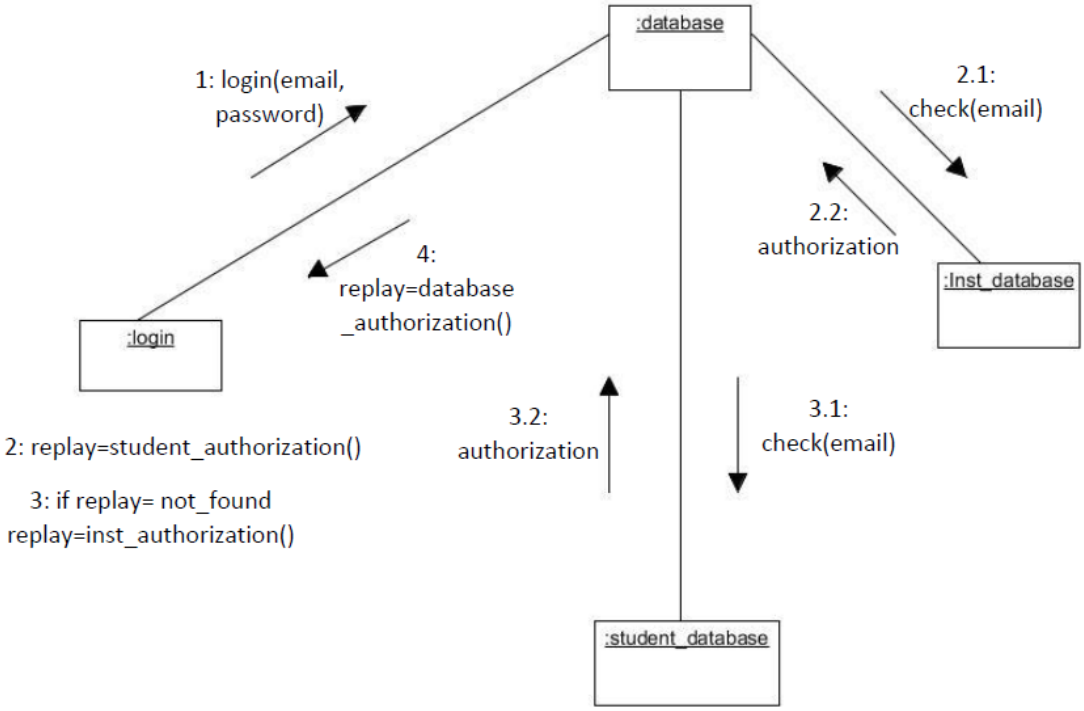



6.1.2 upload file sequence diagram

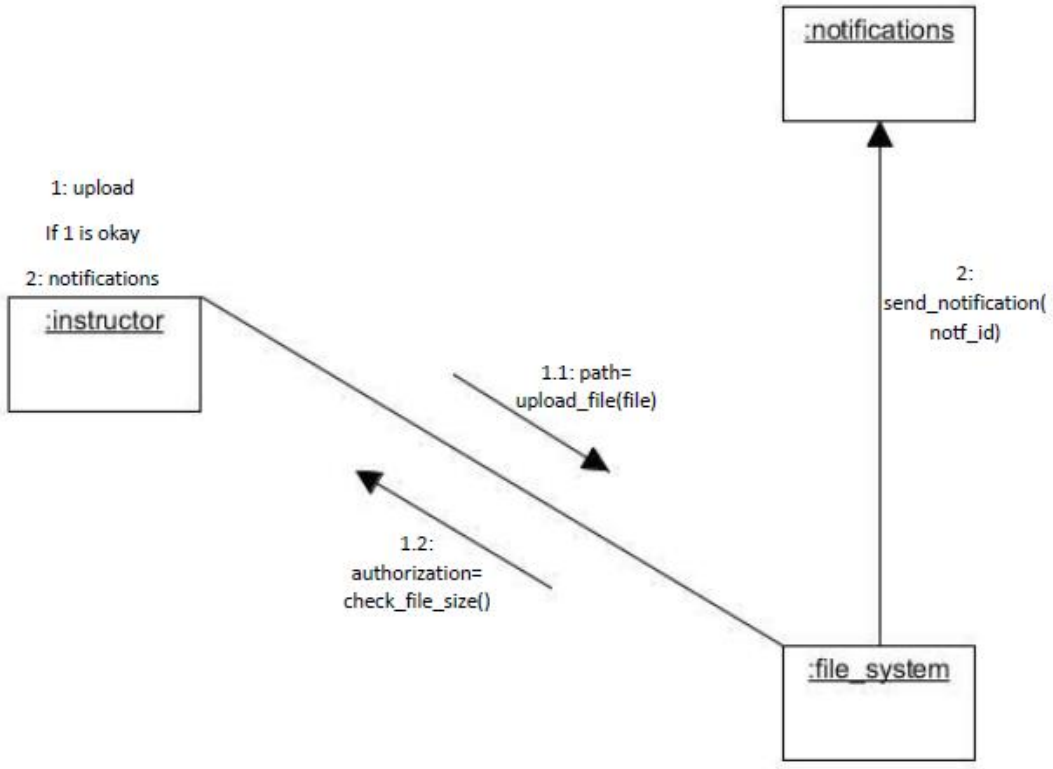



6.1.3 download file sequence diagram

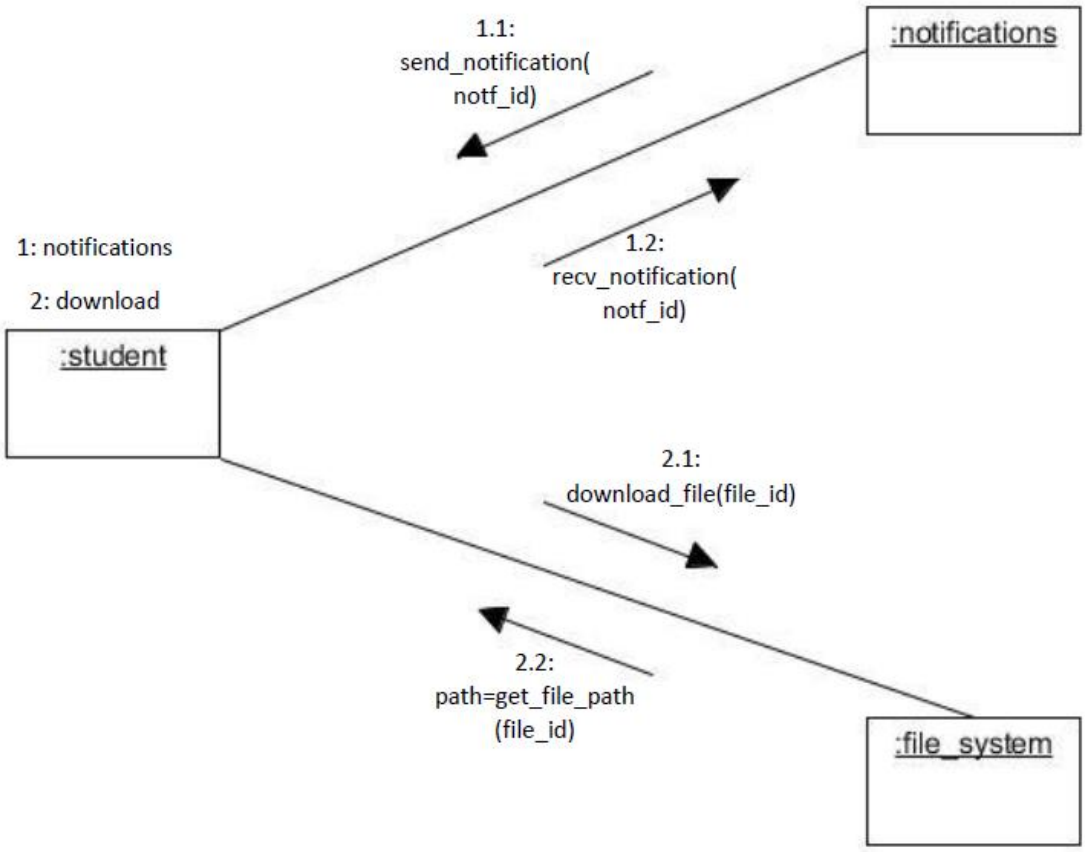
## ❖ Collaboration diagram:




6.2.1 login collaboration diagram

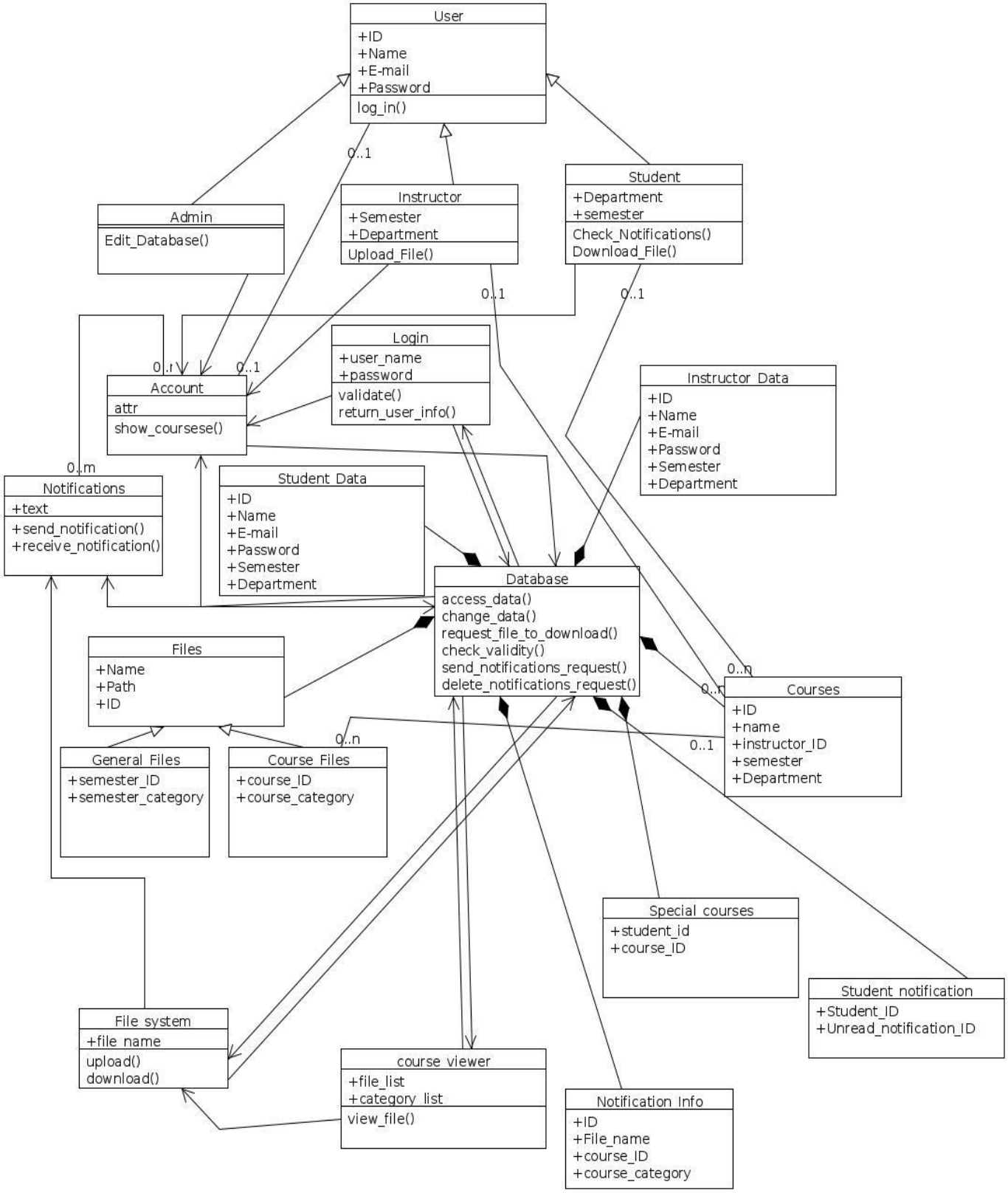



6.2.2 upload file collaboration diagram

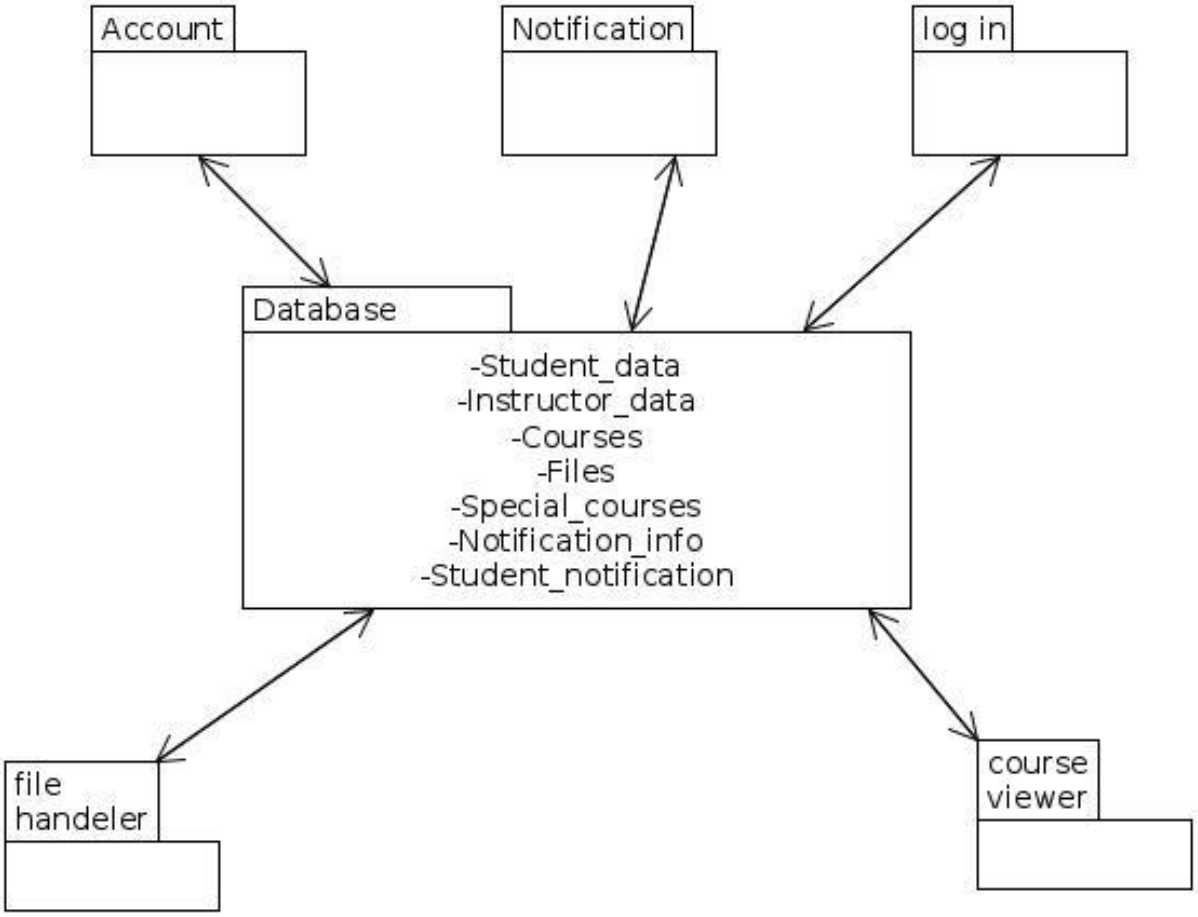



6.2.3 download file collaboration diagram

# Detailed class diagram:



## Architectural model:



## Backlog:

Work with interfaces first to check that this interfaces fits user requirements and then start to implement the functionalities of each interface

| task | priority | state | why |
|---|---|---|---|
| create database tables | 1 | done | all next functions access database so can't make other functions without creating database |
| log in | 2 | done | to test that interfaces access database correctly so we should first implement log in function |
| admin interface | 3 | done | as admin who add instructors and student in database so it should be first interface to be implemented |
| instructor interface | 4 | done | as instructor who upload files in courses so its interface should be implemented before student |

| | | | |
|---|---|---|---|
| student interface | 5 | done | checks that username and his department and courses that he enrolled in them shows correctly |
| insert in database | 6 | done | after the implementation of interfaces start to works with functionalities of each interface as admin who only access database and change in it so his functionalities should be implemented first |
| update database | 7 | done | after inserting in database you can update or delete in it |
| delete database | 8 | done | |
| functionalities of instructor | 9 | done | to make functionalities of student we should first make instructor upload files so student can download it |
| notifications | 10 | done | when instructor upload file system should send notifications to all student that enrolled in this course |

| | | | |
|---|---|---|---|
| functionalities of student | 11 | done | after uploading files and send notifications student can show notification and delete them and show all files in course |
| design | 12 | done | after finishing all functionalities start to work to beautiful design |

## Test cases:

- ✓ Function log in in android:

Input: empty email and password
Output: error message that email can't be empty

Input: empty password
Output: error message that password can't be empty

Input: invalid email
Output: error message that bad format of email

Input: password less than 6 character
Output: error message that password can be less than 6 character

Input: valid email and password but not in database
Output: error message that not found

Input: valid email and password in database
Output: new page that contains true courses which user enrolled in

Input: email and password of instructor
Output: error message that not found

Input: email and password of admin
Output: error message that not found

- ✓ Function log in in web:

Input: empty email and password
Output: error message that email can't be empty

Input: empty password
Output: error message that password can't be empty

Input: invalid email
Output: error message that bad format of email

Input: password less than 6 character
Output: error message that password can be less than 6 character

Input: valid email and password but not in database
Output: error message that not found

Input: valid email and password in database as instructor
Output: redirect to page of instructor

Input: valid email and password in database as admin
Output: redirect to page of admin

Input: email and password of student
Output: error message that not found

Input: name of true table
Output: show true data fields to be filled

Input: insert exist user
Output: error student found

Input: choose semester
Output: list of special courses that not contain courses of this semester

Input: choose department
Output: list of special courses in this department

Input: empty data field
Output: error message that can’t be empty

- ✓ Update database:

Input: not check any option to update by it
Output: error message that you should select an option

Input: check id but id field is empty
Output: error message that id is empty

Input: check email but email field is empty
Output: error message that email is empty

Input: check email but email field isn’t valid as email
Output: error message that invalid email

Input: check id but selected id not found
Output: error message that not found

Input: check email but selected email not found
Output: error message that not found

Input: check id and input found id
Output: new window that contain new fields to update

Input: updated email not valid
Output: error message that invalid email and not update database

Input: updated password less than 6 characters
Output: error message password can’t be less than 6 characters

Input: updated password doesn’t equal confirm password
Output: error message password doesn’t equal confirm password

Input: updated valid password and valid email
Output: alert message that student updated successfully

- ✓ Insert in database:

Input: empty data to be insert
Output: error message that name can’t be empty

Input: fill only name but other fields is empty
Output: error message that email can’t be empty

Input: fill only name and email
Output: error message that password can’t be empty

Input: fill with invalid email
Output: error message that email is invalid

Input: fill only name, email and password
Output: error message that confirm password can’t be empty

Input: password less than 6 character
Output: error message password can’t be less than 6 characters

Input: password doesn’t equal confirm password
Output: error message password doesn’t equal confirm password

Input: insert existing email
Output: error message that email already exist

Input: insert valid name, email, password and confirm password
Output: message that successful student

✓ Insert instructor in database:

Input: empty data to be insert
Output: error message that name can’t be empty

Input: fill only name but other fields is empty
Output: error message that email can’t be empty

Input: fill only name and email
Output: error message that password can’t be empty

Input: fill with invalid email
Output: error message that email is invalid

Input: fill only name, email and password
Output: error message that confirm password can’t be empty

Input: password less than 6 character
Output: error message password can’t be less than 6 characters

Input: password doesn’t equal confirm password
Output: error message password doesn’t equal confirm password

Input: insert existing email
Output: error message that email already exist

Input: insert valid name, email, password and confirm password
Output: message that successful student

✓ Insert course:
Input: insert exist course in the same department
Output: error message that course already exist

Input: insert new course with valid name and department
Output: message that course successfully inserted