LAU
الجامعة اللبنانيّة الأميركيّة
Lebanese American University

Yara Harb – ID: 202001058

Mohamad khalifeh – ID: 202000865

Ali Youssef Solh – ID: 202004405

Rani Salman – ID: 202001002

CSC 490 - Software Engineering

Phase II

Dr. Khaleel Mershad

April 15, 2022

Table of Contents

I- Introduction **3**

II- VLA Context and Architecture **5**

Context Diagram 5

Architecture Diagram 7

III- VLA Interactions **11**

Use Case Diagram 11

IV- VLA Components 13

Contributions 13

Submit Review 14

Upload Note 17

To-do List 22

Track Study Hours 25

V- VLA Structure 29

Class Diagram 29

State Diagram 34

VI- Conclusion 38

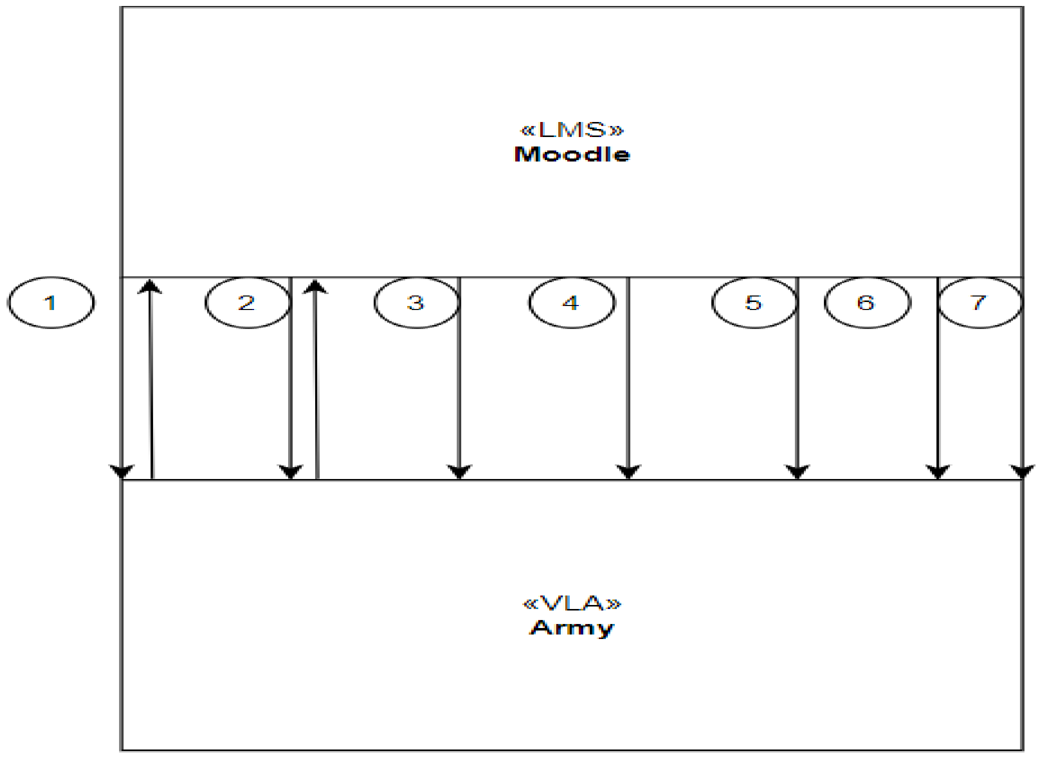
I- Introduction:

The big step after the requirement engineering phase (requirement specification phase) is to design the models of the system. System modeling helps the analyst to understand the functionality of the system, and models are used to communicate with customers. The design process or system modeling focuses on representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). Organized object-oriented design processes involve developing a number of different system models. The first step in OO design is to define the **context** and the **interactions** with the VLA system.This will help the software developer in identifying the different relationships between the VLA and other LMSs subsystems, in addition, the context helps in establishing the boundaries of the VLA. System boundaries help us decide what features are implemented in the system being designed and what features are in other associated systems. In other words, identifying the context helps what features should be included in VLA and what should not be included. This step (context identification) is represented by the **context diagram** that shows that LMS is a subsystem that interacts with the VLA system and different data that both systems exchange. After identifying the context of the system using a context diagram, the interaction between the system and its environment is represented by the use case diagram. The **use case diagram** illustrates the use cases in the VLA and the actors that trigger these use cases. Each use case will be developed at a later stage which enables the software developer to understand the functionalities of the system. Once interactions between the system and its environment have been understood, we as software developers use this information for designing the system architecture. The system architecture is mainly represented using an **architecture diagram**. The architecture diagram shows the main components (subsystems) of the VLA, the packages of VLA, architectural organization of the components and how they are related to each other. It is used to Organize the components using an architectural pattern such as a layered or client-server model. After that, we choose main parts of use cases , we develop **use case scenarios** that will show description , steps and other details regarding the use case. For each use case , we develop an **activity diagram** that shows a sequence of activities and steps related to this use case. The next step is to identify the **object classes** in the VLA. This is done by scanning the requirement document and use cases scenarios previously implemented and highlighting the verbs (methods or behavior) and nouns (classes). After identifying the object classes, we now can develop **a sequence diagram**. The main difference between the sequence and activity diagram is that the sequence diagram shows more technical details and shows sequence of method calls not activities. The sequence and activity diagrams represent the behavioral model of the VLA system. On the other hand, the structural model of the VLA system is represented by the **class diagram.** The class diagram is implemented based on classes previously identified and sequence of methods in the sequence diagram. Thus the class diagram shows the different object classes of the system and relationships between these classes. The development of the design models also include a state diagram. After examining the class diagram previously implemented, the state diagram is developed. The **state diagram** shows the various states that the VLA could be in and the triggers (events) that lead to translation from one state to another. This is also important since it depicts the behavior of the system. They are used to show how objects respond to different interactions or requests and the state transitions triggered by these interactions/requests.

The last step in the design phase is to identify the **object interfaces.** These interfaces act as templates for the object classes and force the class implementing these interfaces to include the methods or behaviors that the interface provides. In general, we represent different design models(context , interaction , behavioral and structural) in this phase using different forms of diagrams. The context diagram, architectural diagram and the class diagram are considered as static design models. As for the use case diagram, the activity, sequence and state diagram are considered as dynamic design models.

II- VLA Context and Architecture

A. Context Diagram:



According to the system requirements in the requirements engineering document, requirements UNF02, SFS06, SFS03, SNFC01, and SNFC02 show that the environment in which our VLA operates consists only of two systems: the VLA itself and the LMS (subsystem) to which it is connected.

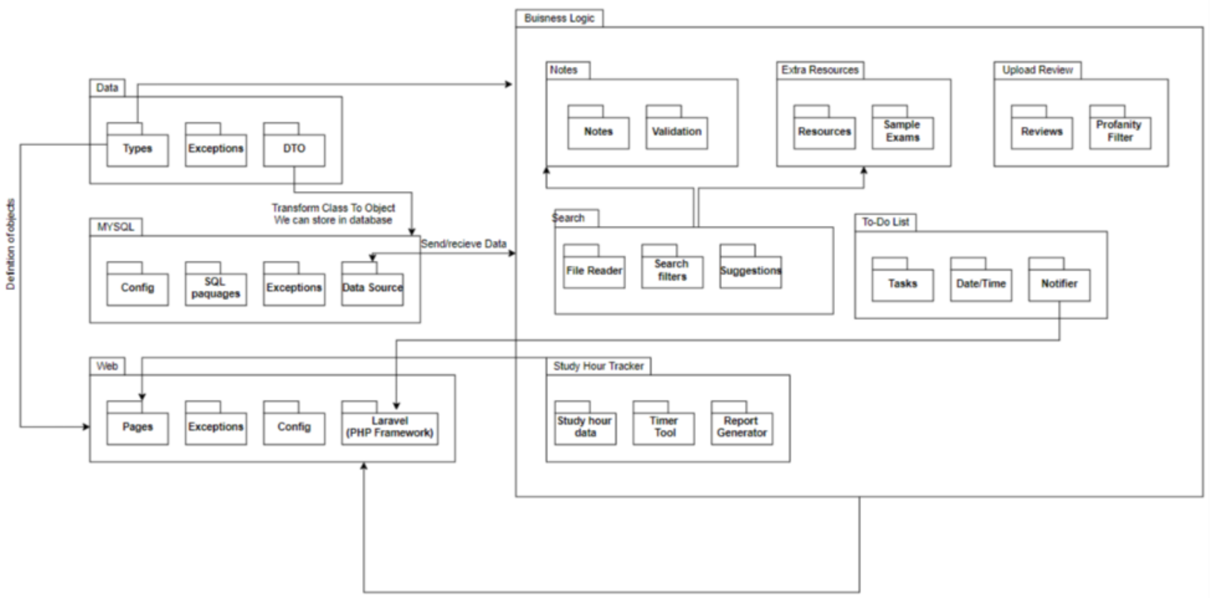
We noticed 6 main links between the two systems:
1- The VLA has a feature that tracks the study time of the student using a stopwatch that students start and stop when they enter and leave the VLA. When the student finishes studying, the VLA should send the study time to the LMS to let the instructor access it. Later on, for each student the system will generate a report , using predefined instructions(such as grades and average of the class) on what to include in the report,and return it to the VLA. The VLA enables the

instructor to read the report and the instructor will analyze the progress of each student.

2- Each student will receive notifications from the LMS based on the frequency preference that he sets in the VLA and the tasks that he puts in the to-do list, in addition to any notifications that the instructor decides to send to his in case he was late in his study plan or in case his notes were not accepted to be published….
To create these notifications, the VLA will have to send to the LMS the content of the notification and the frequency that the student has previously set to receive these notifications, and the LMS would be responsible to send these notifications to the users accordingly.

The LMS provides a list of web services to LMS to organize its functions:
3- The LMS provides the VLA with the class roster, which represents the students that are enrolled in each course provided.
4- The LMS provides the VLA with the list of courses that each student is enrolled in.
5- The LMS provides the VLA with the courses that each instructor teaches.
6- The LMS provides the VLA with the Course resources and documents available for every course taught, so the students can access and study them.
7- The LMS provides the VLA with the progress of each student on his tasks and assignments to keep track of his progress.

B. Architecture Diagram:



Once the interactions between the system and environment is specified, the system architecture can then be deduced.
In the architecture diagram, we identify the major components or subsystems that make up the system and their interactions. For each objective, a subsystem was created, and then all subsystems were combined into a more general subsystem called “Business Logic”. In addition to the Business Logic, the architecture diagram includes MySQL (for the database), Data (for the data saved), and web (for the user’s display and UI). The connection between two subsystems or two packages indicate that this subsystem or this package interacts or shares information with the other subsystem or package.

To begin with the Business Logic, according to the system requirements SFI08, SFS12, SFS13 and SFS15, the first component that should be added is the “Notes” which includes the following packages: Notes and Validation. In the Notes package, the notes of the students that they upload whether it is private or public is included. Moreover, the Validation package contains the validation of the instructor in case the student decides to publish his/her notes publically.

For the Extra Resources subsystem, according to the requirements SFI05 ,SFI06, SFI03 and SFI04 in the requirement engineering document , it is made up of “Resources package” which contains the resources that are uploaded by the instructor in the aim of helping the students in his/her studies and Sample Exam package which includes the sample exams posted by the instructor formed of multiple choice questions that will then be corrected automatically by the system using the answers provided by the instructor.
In addition, according to requirements SFS12,SFS15, SFS14, SFSI01, and SFSI02 we have the “Search subsystem” that includes the “File Reader package” which will go through the files that are in the database of the system and reads its data and Search filters package which allows the students to search in the desired section (notes, in the title of the documents, in the context of the documents, etc.). Furthermore, the Search subsystem includes a “Suggestions package” which will give the student the result found as a consequence of the search.
The Search Subsystem is then connected to the Extra Resources and the Notes subsystem to be able to access the notes, resources, and sample exam in the search.

According to the requirements SFS09,SFS10, SFS11, SFS14,SNFPR01,SNFA02 and SFA01, the Business Logic also has “Upload Review” subsystem which in its turn is made up of the Reviews package which handles the reviews of the students for the instructor and the Profanity Filter package that will spot the inappropriate review of the students that violates the code of conduct.
Another component included in the Business Logic is the “To-do list” according to requirements SFS04,SFS05, SFS06 and SFS07. This subsystem is made of the Tasks package that consists of the tasks that students have and decides to complete and the Data/time package that allows the students to specify the deadline of the task. The To-do list also contains the “Notify package” that is connected to the Laravel package in the “Web” since the notification feature is integrated from the Moodle LMS. This package will notify the students of the task when it is the time of the deadline.

Finally, system requirements SFS02, SFI02 and SFSEC01 show that the last subsystem included in the Business Logic is the “Track Study Hours”. In this subsystem, it contains the “Study Hour Data package” that handles the number of study hours, the Timer tool that measures the study hours, and lastly Report

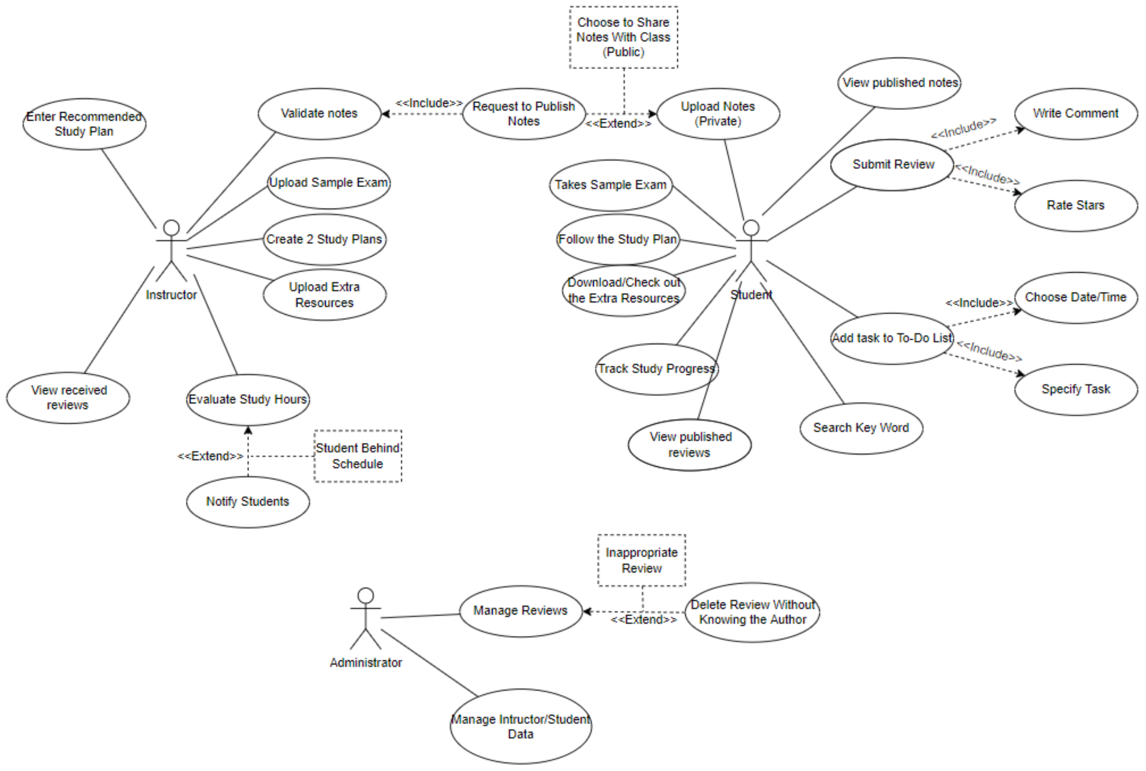
Generator which gives a report of the hours studied for each student and a comparison with the required study hours given by the instructor. The whole subsystem “Track Study Hours” is connected to the Web as the study hours are calculated from the time spent on the web-page using the stopwatch.
At last, the whole Business Logic is then connected to the Web subsystem (system requirement NSFD01), since all activities are done on the web and the web will display those activities.
Outside the Business Logic, since the reviews, notes, courses and others will be stored in the database, we have a “MySQL” subsystem, which basically is the database of the system. The first package that this subsystem contains is the Config package, which handles the configuration of the database. Another package that is present is the SQL packages that contain the components needed to use MySQL. MySQL also contains an Exception package that will handle an error that occurs with the data whether it is invalid or if it has invalid datatype. Finally, the last package is the Data Source package that includes the data of the whole system stored in the database in tables. The Data Source is connected to the Business Logic because there will be send and receive data between them.
The Data package is another subsystem implemented in the VLA, this subsystem handles organizing the data and classes that are used in the system. These subsystems contain the “types” package, which contains the classes and attributes to each of these classes in the system. This package is linked to the “web” and “logic” subsystems and defines the objects in these subsystems. For example, In the “types” package, The instructor is defined as a class that contains specific attributes, When we want to display the instructor info on the web page, we would use the “types” attribute to extract the attribute of the class instructor and display them, so it would like the instructor has a name, email, ID, list of courses he teaches …
The “types” package is used in the same way in the “Business logic” subsystems as it would determine the class “Note” for example and specify the attributes of this class to be the content of the note, the date it has been uploaded…
The “exceptions” package is another package in the “data” subsystems, as it would specify the boundaries or limits for each type of data, and it would also include the actions that will be performed if one of these boundaries have been violated on the server side of the system.

Finally, we have the “DTO” package in the “Data” subsystem, this package contains the instructions on how to transform the data contained in the “data” subsystems to a format that can be saved in the database. For this reason, we linked this package to the SQL package to transform the class objects to a format that we can store in the SQL database subsystem. The “exceptions” package contains the

The Final subsystem in the “web” subsystem, this subsystem is responsible for the user interface of the users, it is the interface with which the user interact with the other subsystems like retrieving and saving data in the SQL database, or view the notes or reviews that are present in the business logic database. This subsystem first contains the “pages” package. This package contains each page in the system that the user interacts with, like the “submit review” where the student can go to select a course than an instructor to submit his review on this instructor page, or the “extra resources” page where the student can see the resources posted by the instructor. The next package is the “exceptions” package, this package contains the types of exceptions that would be thrown if the user violated one of the boundaries specified for each section. This package is responsible for keeping the user from inputting incorrect data from the user level of the system, as it would show the user error messages or stop him from clicking on the submit button or any other kind of restrictions on the user level. This package is also responsible for handling any other kind of exception like entering a non-existing web page or retrieving data that is not present in the system … The “config” page in the “web” package is responsible for customizing the way our site or a specific directory on your site behaves.
The “Laravel PHP framework” is an add-on extension to make KoolReport work seamlessly in the Laravel Framework environment. By adding a package, a report created with KoolReport will recognize Laravel databases automatically. Furthermore, KoolReport's widgets will be configured with an assets path and URL so that all are working without requiring any further set-up effort from you.

III- VLA Interactions

Use Case Diagram:



We created a Use Case Diagram to illustrate our system's interactions with its environment. The previous diagram contains three different actors: students, instructors, and administrators. We will now describe the various ways these actors may interact with our system.

First of all, our system contains utility features that help increase students' productivity. A student can organize his/her To-do list by adding tasks to it. Creating tasks includes the student entering the task description and selecting a date and time which will represent the deadline of the task. Of course, the student can mark their tasks as complete whenever they complete them. When the deadline of a task approaches the student will be reminded by the system to work on the task with a notification.

Another utility feature of the system is that the user can organize his notes by uploading them to the system. The student will first select a course that he/she is enrolled in to keep their notes document separated and organized. The student may select a document from their device to upload to the system and the student will have to select between uploading the document as a private or public note. If the user selected the notes document to be private then the document will be uploaded and stored in the system process will end there. However, if the user chose the public option then there are more steps to the interaction. The instructor will receive the uploaded document to check if the notes are correct and good to share with the rest of the class. If the instructor finds that the notes have some mistakes then they will write a comment to the student, criticizing and correcting their notes. However, if the instructor found that the notes are good then they will be published to the system for all students in the same course to see them. This is another way that the student could interact with the system; the student can always view public notes that have been posted and approved by the students and instructor of the same course.

Another feature that can help the student is the search bar. If the student is lost and cannot find a certain document previously posted by their instructor, then they may use the search bar to type in a key work and then they will obtain results that will guide them towards what they are looking for.

A student may also keep track of his/her progress by viewing the assignments and tasks that they completed recently. A student may also view the number of hours that they are spending studying at the end of each week to evaluate their own productivity. An instructor may also do the same. Each instructor has access to study hour reports of all their students, and in the case where they find one of their students behind schedule then they can send a notification checking up on a student and encouraging them to pick up their work. The instructor will evaluate the student’s work by comparing it with the recommended study plan that they post for the students to follow and organize their work accordingly.

The system also provides features that will help students prepare for exams. The instructor may upload extra resources for the students to consult, view, and download. The instructor could also create sample tests in the form of MCQs for the students to practice on. The instructor also has the capabilities to create different study plans that fit the level of knowledge of each student. For example, a student that has trouble with the material will receive a study plan that is more intensive than others. As for more capable students, they will receive a lighter study plan.

Another important way the users could interact with the system is through the review section. Students could create and post anonymous reviews about their current or previous instructors. This process includes the student writing his/her or opinion on the instructor in a textual form. Then, the student will be able to rate their instructor skills by choosing ratings between 1 and 5 stars for different criterias of teaching. Students can view reviews of all instructors in the review section. This will help them evaluate which instructor to register a course with. An instructor may also take a look at reviews about them to obtain feedback and comments on their work. An important thing to not leave out is that before each review is published, it is evaluated by a system administrator to check if the review only contains constructive criticism and feedback and no hateful speech. If the review is inappropriate the administrator may delete it and not allow it to be published. The other type of interaction that an administrator can do, (which is a natural privilege of administrators) is that they could manage the system data.

IV- VLA Components

Contributions

We all worked on all the use cases but since you asked to specify each use case to one of the group members, we assigned the use case to the member that worked on it the most.

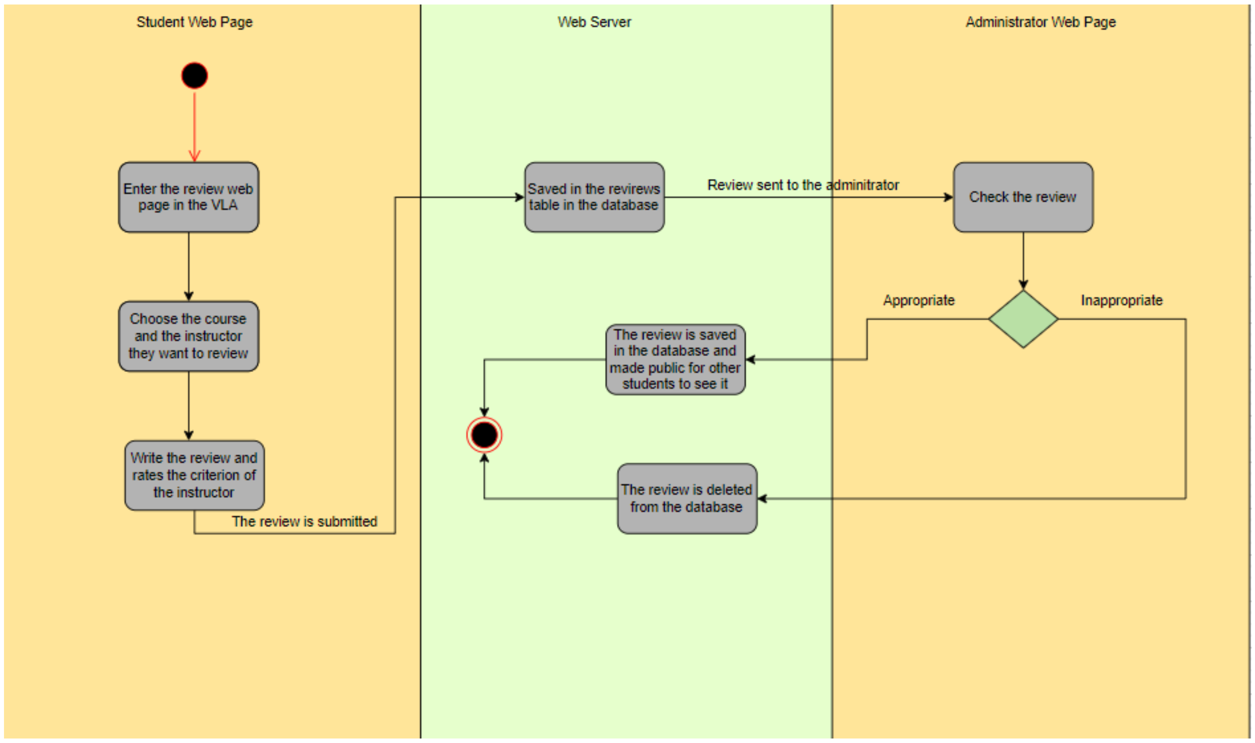
Ali Youssef Solh	Submit review
Rani Salman	Track study hours
Mohamad khalifeh	Upload note
Yara Harb	To-do list

A. Submit Review Use Case

a. Use case Scenario

Use case name: Create Review	
Area: Review Section of the page	
Actors: Student, Administrator, Instructor	
Description: Students write a review on their instructor	
Stakeholder: Student, Instructor	
Level: Pink	
Triggering Event: Student submits a review	
Trigger Type: [✓ **External**] **Temporal**	
Steps Performed	**Information for Steps**
1- The student enters the review web page of the VLA (where student can create or view other reviews)	Review Web Page
2- The Student chooses to create a review	Review Web Page
3- The System runs a select SQL query to extract the student's currently/previously enrolled courses	Course List of student
4- The student selects the course he/she wants to write a review on	Student's input (selected course)
5- The review will be about the instructor that taught the student the previously selected course (this is an automatic process since every course instance in the Course table in the database has one main instructor)	Instructor is chosen
6- The student can now write his/her review of the instructor	Student's input (written paragraph)
7- After writing the review the student will rate the instructor skills using a scale between 1 and 5 stars	Student's input (rating)
8- The Student finalizes his review and presses the submit button	Awaiting student's decision
9- The review written by the student is sent from the user's device to the VLA server	
10- The review is saved in the reviews table of the database, and the review isn't published yet	Student's review
11- The administrator will now evaluate the review	Administrator's decision
11.1 If the review is inappropriate, the administrator deletes it from the reviews database table	Code of conduct
11.2 If the review is appropriate, the administrator will be approve the publishing of the review, and changes its state from pending for approval to published	
12- Depending on the administrator's decision the review will or won't be published to the reviews web page	
Preconditions: Student should be enrolled in a course with this instructor or has previously taken the course with the instructor	
Postconditions: Review is published	
Assumptions: The review respects the code of conduct	
Success Guarantee: Respectful review is published or the review got deleted	
Minimum Guarantee: The student was able to write and edit his review	
Objectives Met: The review is published	
Outstanding Issues: How to handle the review if it was disrespectful or violates the code of conduct?	

b. Activity Diagram

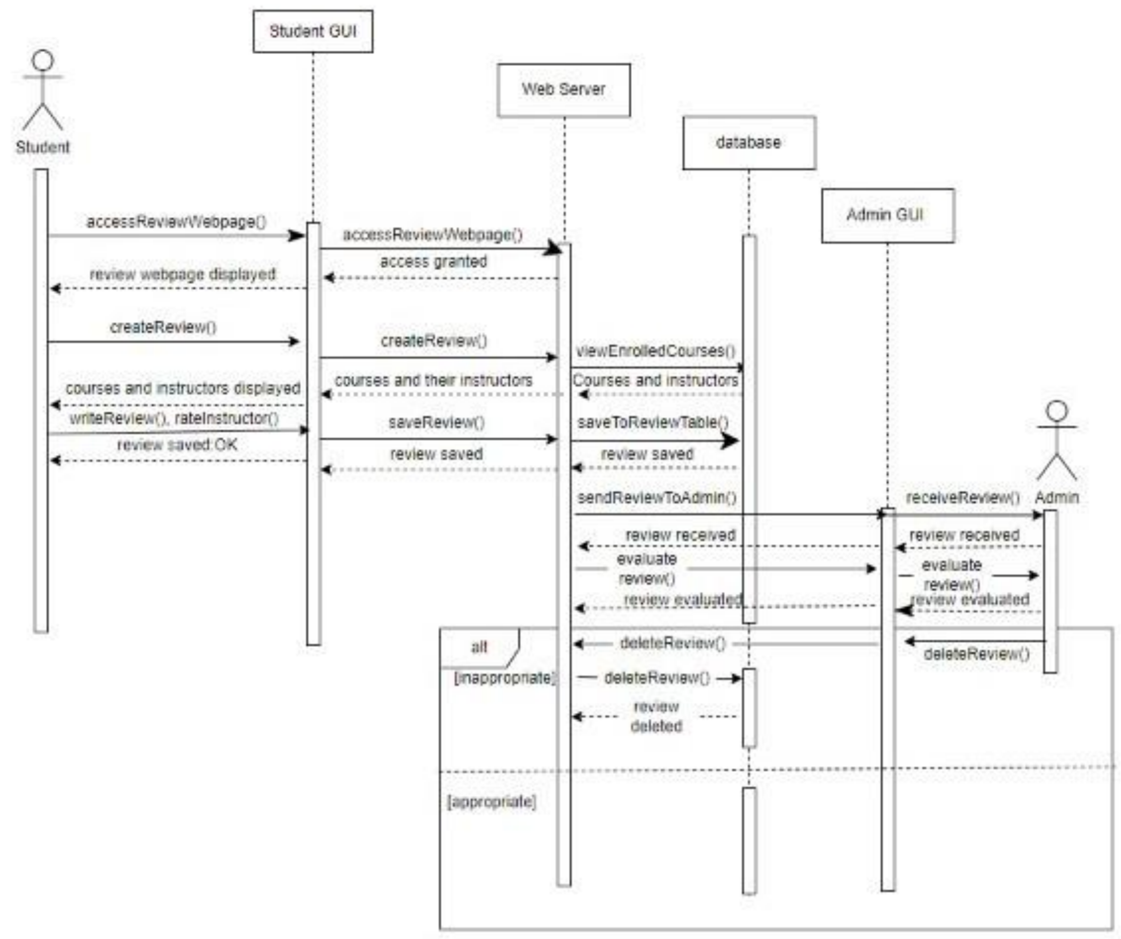

Student Web Page
Web Server
Administrator Web Page
Enter the review web page in the VLA
Choose the course and the instructor they want to review
Write the review and rates the criterion of the instructor
The review is submitted
Saved in the reviews table in the database
Review sent to the admnitrator
Check the review
Appropriate
Inappropriate
The review is saved in the database and made public for other students to see it
The review is deleted from the database

The activity diagram consists of 3 different platforms which are student web page , web server and Administrator web page. Thus , the activity diagram consists of 3 swimlanes.
The start point is when the student enters the review web page in the VLA (1st activity) then the student will choose the course and the instructor who will be evaluated (2nd activity). After that the student will write the review about the instructor chosen and add rating by stars_slider (3rd activity). Here the web server will save the review to the database (4th activity) and send it to the administrator who in turn will check if this review is valid (5th activity). If (decision here) the review is valid then the review will be published by the server and other students can see it (6th activity). Otherwise, the review will be deleted from the database (last activity). By these activities we reach the end of this use case.

c. Sequence diagram

In order to create the sequence diagram it is necessary to go through the use case scenario and highlight the verbs and nouns that are possible methods, classes, or attributes. After we divided them we collected the nouns and verbs and started with the sequence diagram.

In italic: methods (action, verb)
In bold: classes (classes or attributes in the classes, nouns)



The classes are: Student GUI, Web server (ReviewHandler), Admin GUI, and Database.
The actor involved: Student and Administrator
To begin with, the student will first access the review list page by pressing on a button and in its turn the Student GUI will then call the method

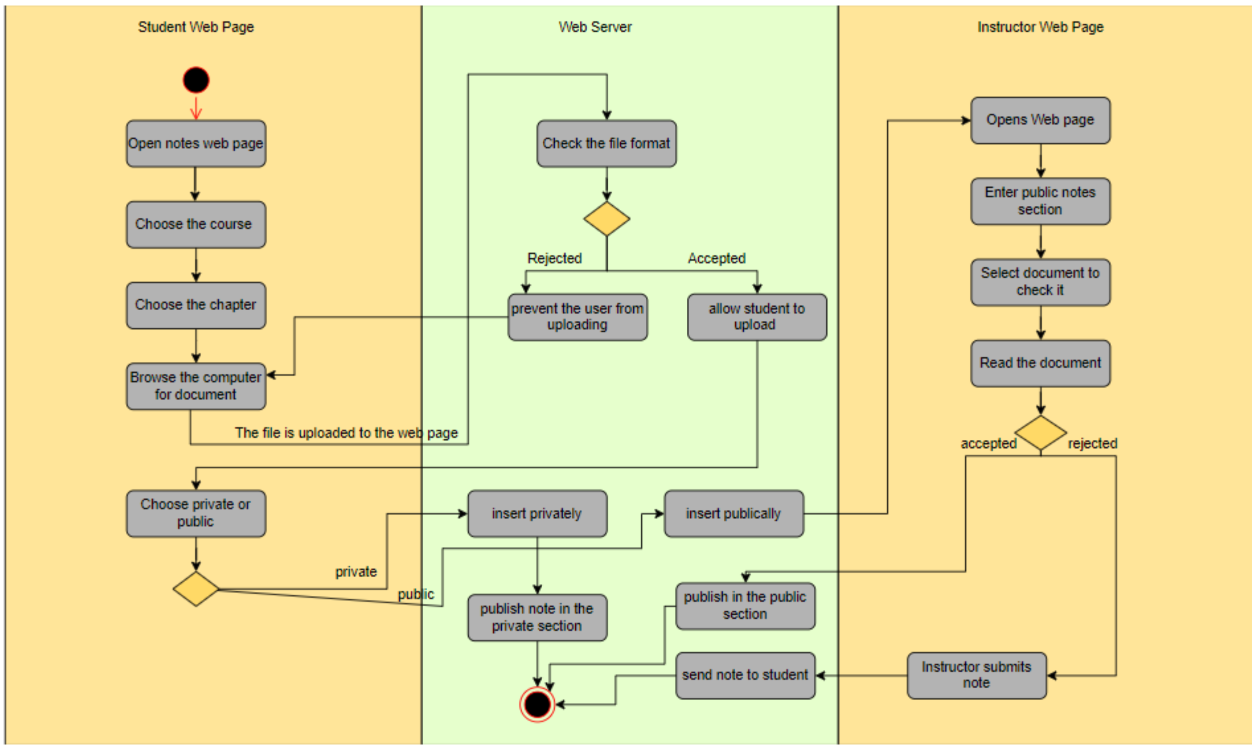
accessReviewWebPage() in the Web Server. So the student GUI class will be an intermediary between the Student and Web server and the admin GUI class will be another intermediary between the admin and the Web Server. After the server gives access to the review list page it will return a message “access granted” to the student GUI class and then the student GUI class return to the student message “review web page displayed”. Afterwards the student will request to write a review through the Student GUI class (createReview()). The student GUI will send the review request to the Web Server by calling createReview() methods. Then the web server will call the database to display the list of enrolled courses and instructors. After this query, the student GUI will display the list of the course and instructor to the student .Now, the student will choose the course and instructor and write the review and rate the instructor using the writeReview() and rateInstructor(). Once the student is done with writing the review and rating the instructor, the review will then be sent to the server to be saved by calling the method saveReview() from the student GUI. The Web Server will call the method saveToReviewTable() in Database in order to save the reviews in the database. At last, the review will be saved and a return message (to confirm the saving) will be sent from Database to Web server, from Web server to Student GUI, and from Student GUI to Student (“review saved”). The system now (server) will send the review to the admin. The admin sees the review through the Admin GUI after it has been saved in the database (receiveReview()) with a return message “ review received” .The next step is to evaluate the review in which the server will request from the admin to evaluate the review (evaluatereview()). After the admin evaluates the review and sends a return message “review evaluated” for the Admin GUI and the server. Now we have two alternative paths whether the review will be either accepted or deleted by the admin according to the code of conduct. In case the review was inappropriate and violates code of conduct, the Admin GUI will call the deleteReview() method in the Web Server. The server will then call the removeReview() method in the Database where the Database will delete the review and send a confirmation message to the Web server that it is deleted (“review deleted”). Finally, in case the review is appropriate the admin will leave the review and keep it in the database.

B. Upload Note Use Case

a. Use case Scenario

Actors: instructor and student	
Description: The student upload document to VLA	
Stakeholder: instructor, student uploading his notes and all students taking the course	
Level: Red	
Triggering Event: Student chooses to upload document to VLA	
Trigger Type: [✓ ■ **External**] ☐ **Temporal**	
<u>Steps Performed</u>	**<u>Information for Steps</u>**
13- The **student** *enters the notes section* of the VLA	Notes Section
14- The student *selects the* ***course*** he wants to upload his **notes** to	Course List
15- The student *selects the* ***chapter*** he wants to upload the note to	Chapters list
16- The student *browses his/her computer* and selects **a document** to upload to the web page	Files list
17- The ***system*** *checks if the uploaded document's format is accepted* (pdf, docx, …)	The chosen document
5.1 If the document is reject the system *will not let the* user submit the document The user will then have to *choose another document* to upload	The student's notes
5.2 If the document is accepted, the system will allow the student to *upload his notes to the web server.*	The student's notes
18- The student *chooses* **the private or the public option** from the web page before finally uploading	
19- The document is *sent from the student web page to the web server.*	The chosen document
20- The VLA server will *insert the notes* to the database table.	The chosen document
8.1 If the student wants to make his notes private, the server saves the notes in database and *restrict the view to the student who uploaded the notes only.*	The chosen document and the student decision private/public
8.1.1 The **web system** will *publish the note privately*	
8.2If the student wants to make his notes public, the server send *these notes to the instructor to get his* ***approval*** before publishing the notes	The chosen document and the student decision private/public
21- The instructor *enters the notes section* in VLA	
22- In a specific section of the page the instructor will *see the documents that are pending his approval*	List of notes pending for approval
23- The *instructor selects which document he want to view and opens it*	List of notes pending for approval
24- After reading the document the instructor either *approves the document or rejects it*	The chosen document
12.1- If the notes are accepted, the server *receives the approval* from the instructor and they will *be published in the public notes section*	The chosen document
12.2- If the notes are rejected, the *server gets the response from the instructor and return the note to the student.*	The chosen document
Preconditions: The student have written notes and all the students should be enrolled in this course and the instructor should be teaching this course	
Postconditions: The notes are uploaded to the VLA	
Assumptions: the notes should be consistent with the explained materials.	
Success Guarantee: Public notes are correct or private notes are uploaded	
Minimum Guarantee: the student was able to upload the notes with the option to upload them privately or publically	
Objectives Met: All students can publish notes	

b. Activity Diagram

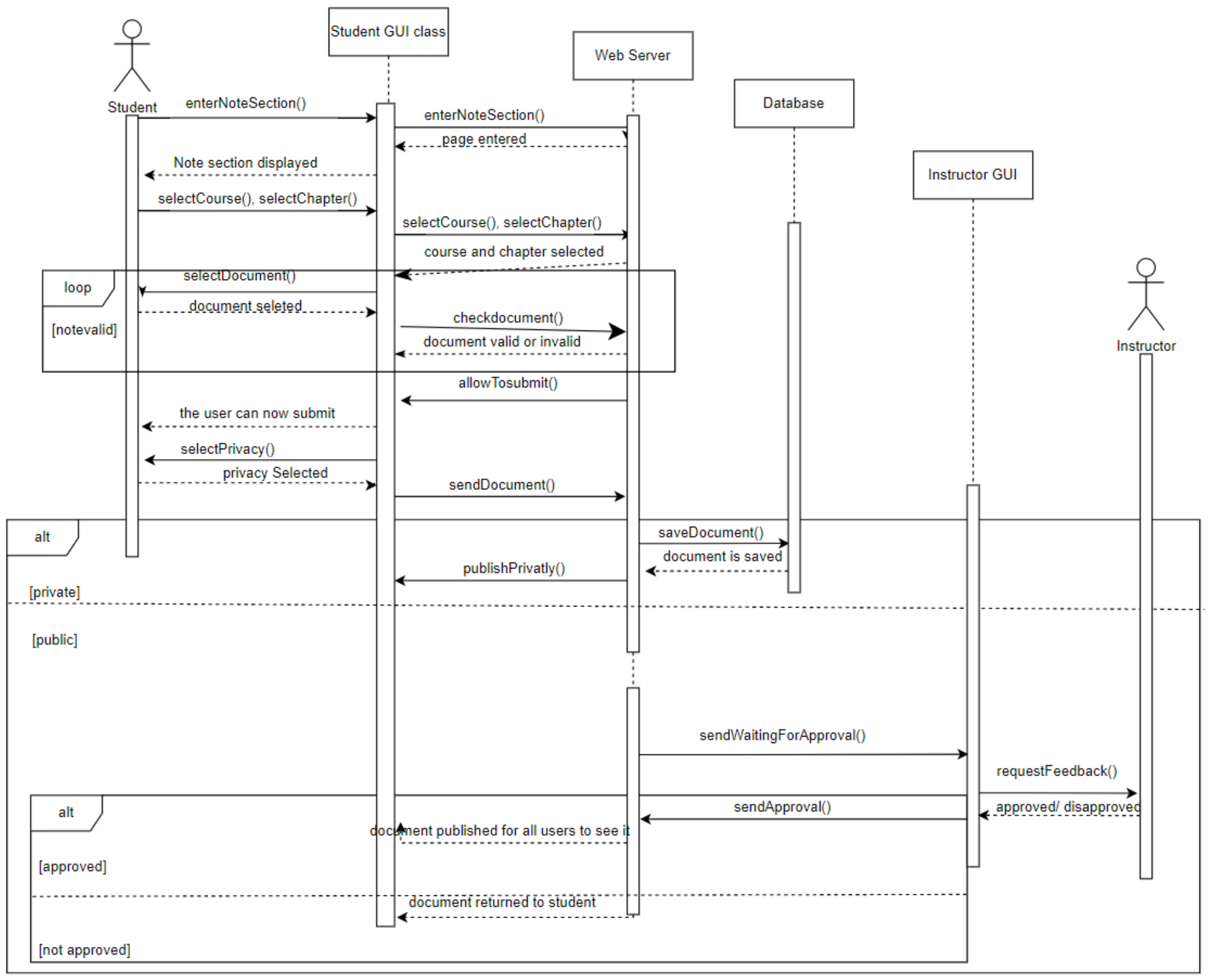


The activity diagram consists of 3 different platforms which are student web page , web server and Instructor web page. Thus , the activity diagram consists of 3 swimlanes.

The start point is when the student enters the notes web page in the VLA(1st activity) then the student will choose the course and the chapter he/she wants to upload his notes to . After the student will browse the device for the document, ten web servers will check the format of the document. If it is accepted, the student will choose to publish its note either as private notes or public notes. If he/she chooses to publish them privately , the web server will add the notes to the database and then terminate. However, if the student wants his notes to be public , the web server inserts the notes to the database and sends them to the instructor who in turn will enter public notes and read the document. If the note is accepted, the web server will publish the notes and other students can see it. Otherwise, the instructor will resend the note to the student and the program terminates.

On the other hand if the file format is not accepted, the server will request from the student to change the format of the document.

c. Sequence Diagram



Classes are: Student GUI, Web Server(Note Handler), Database, and Instructor GUI

Actors involved: Student and Instructor

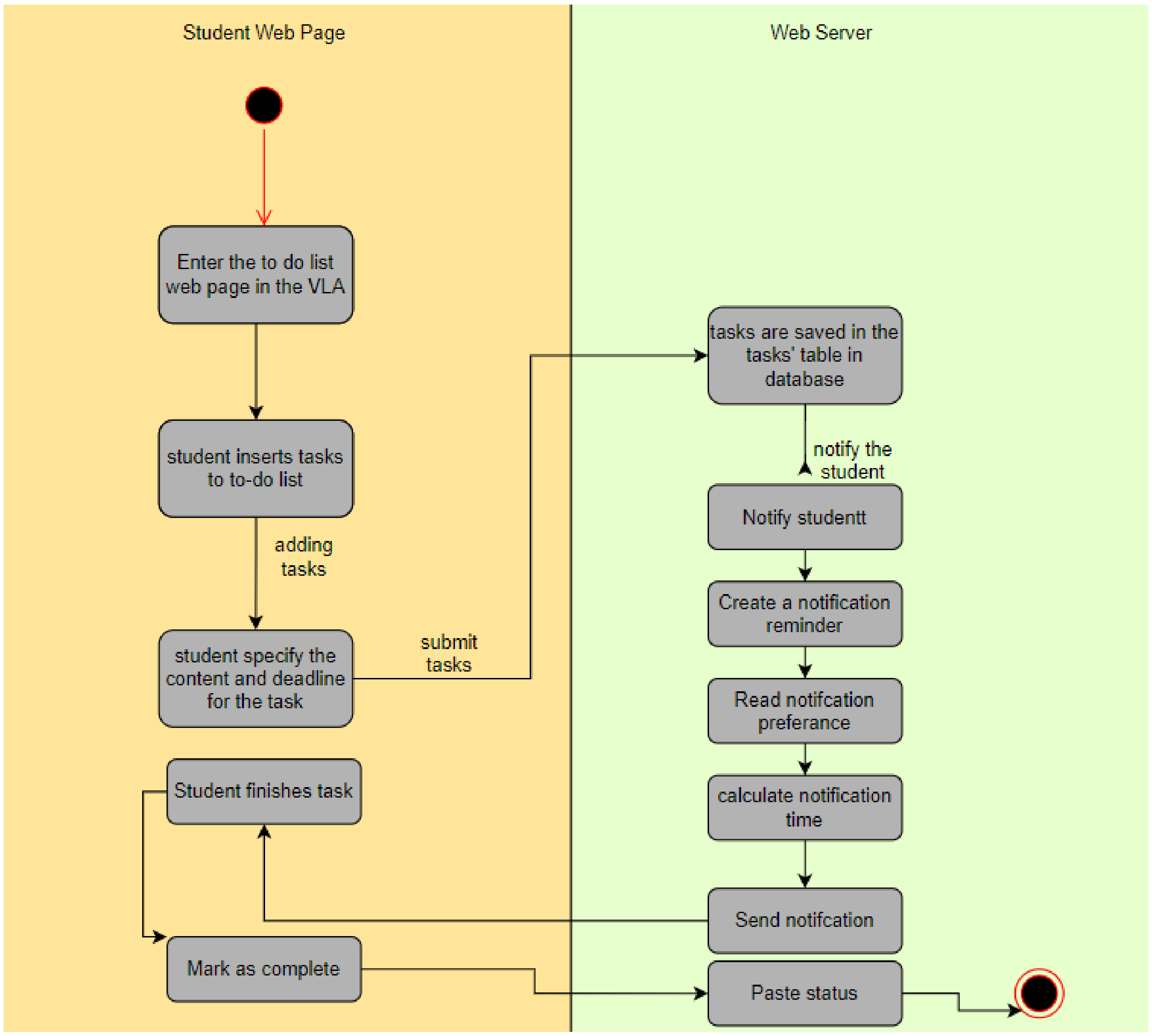
To begin, the student will first enter the notes web-page by pressing on a button and in its turn the Student GUI class will then call the method enterNoteSection() in the Web Server. The GUI class will be an intermediary between the Student and Web server. Subsequently, the Web Server will return "page entered message" to the GUI the class and then the GUI class will return "notes section displayed" to the student. When a student enters the page, they will decide which course and which chapter they want to upload their notes document to. This will call the selectChapter() and selectCourse() methods of the GUI class which then communicate the select course and chapter to the server using the selectChapter() and selectCourse() methods. The Web server will then return "course and chapter selected” message to the GUI. We will then enter a loop. The Student will select a document from their device to upload. The GUI class calls the selectDocument() method and returns the selected document to Web Server. The Web Server calls the checkDocument() method and returns if the document is valid or not. If the document is not valid we repeat the loop. If the document is valid we exit the loop, the Web Server calls the allowToSubmit() methods indicating to the user that the user can submit/upload their document. But before submitting, the student selects the type of privacy of the document through the selectPrivacy() method of GUI class. The previous method then returns the selected privacy. The document will then be sent to the Web Server (using sendDocument()). The Web Server class will then save the document to the Database using saveDocument() method. This method will then return that the document is saved upon success. We arrive now at the alt section of the diagram. Depending on the privacy mode selected, we either enter the private section or public section. If the student chose the document to be uploaded privately, then the Web Server will call the publishPrivately() method, and this is the end of the use case if the private mode was chosen. Now, if the student chooses the public mode of uploading, the Web Server will call the sendWaitingForApproval(), which will communicate to the GUI class of the instructor to display the document and request the instructor feedback on the notes. The GUI class of the instructor will do this by calling the requestFeedback() method. This method will return if the instructor approves or disapproves the note document. The instructor GUI class will call the sendApproval() method which will send the decision of the instructor to the Web Server. The Web Server will act upon this information and return a result accordingly. If the instructor approves, then the document will be published for all students enrolled in the same course to see. However, if the instructor disapproves the document will not be published.

C. To-do List Use Case

a. Use case Scenario

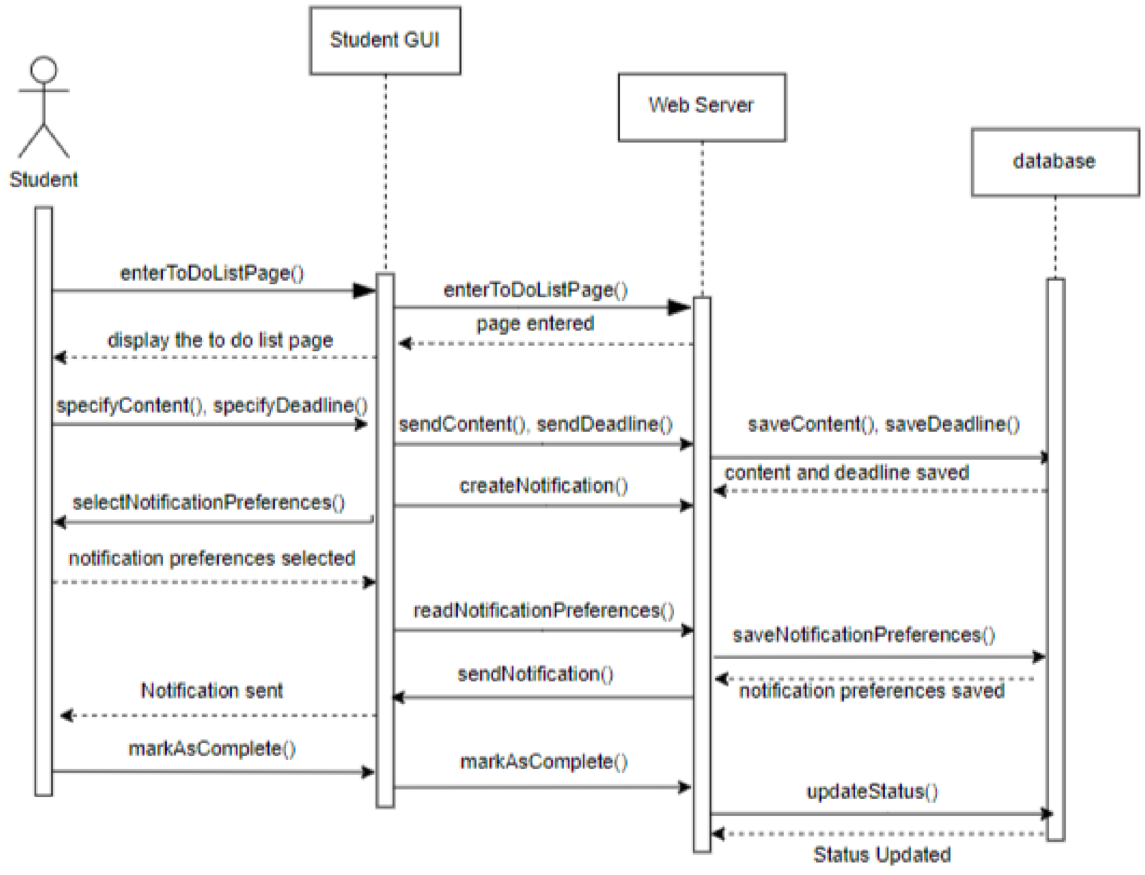
Use case name: To-Do list	
Area: To-Do list Web page	
Actors: Student	
Description: The student puts the tasks that he wants to complete in a to-do list format	
Stakeholder: The students	
Level: Blue	
Triggering Event: The Student enters the to-do list web page and enters the tasks	
Trigger Type: [✓ **External**] **Temporal**	
<u>Steps Performed</u>	**<u>Information for Steps</u>**
1. The **student** *enters the to-do list web page*	To-do list web page
2. The student *starts to insert his task* in the to-do list	The student tasks
3. For each **task** the student *specifies* the **content** and the **deadline** to their task	The tasks' content and deadline
4. The student *submits this task* and the task will *be send to the **VLA server*** to *be saved there*.	
5. The VLA *server saves this task* in the **database** saving the content and the date/time	The Task submitted by the student
6. The systems *creates a notification reminder* in the Tasks table in the database. The **notification** reminder contains the deadline of the task.	
7. While the system is running, *it reads the data from the* Tasks database every 1 minute.	Data from the tasks database
8. The system *reads the notification preferences* of the user and sends them to the database.	User notification Schedule
9. The database then *saves the notification preference*.	User notification preference
10. If the notification time of a task is now, the system *sends a notification to the student* by calling the notification web service in the LMS.	
11. When the student finishes a task, he will *mark it as completed* in the to-do list	The status of the task
12. When the *student changes* **the status of the task** on the web page, the web *server changes its status in the database* to "complete"	
Preconditions: The student has many tasks to accomplish	
Postconditions: the student sticks to the to-do list and accomplish the tasks	
Assumptions: the student is willing to stick to the to-do plan to study	
Success Guarantee: the student will be able to organize their tasks and accomplish them	
Minimum Guarantee: The student can put his tasks in the to-do list web page	
Objectives Met: The student is following his schedule and accomplishing his tasks in an organized manner	
Outstanding Issues: What happens if the student makes a to-do list but doesn't do the tasks?	

b. Activity Diagram


Student Web Page
Web Server
Enter the to do list web page in the VLA
student inserts tasks to to-do list
adding tasks
student specify the content and deadline for the task
submit tasks
tasks are saved in the tasks' table in database
notify the student
Notify studentt
Create a notification reminder
Read notifcation preferance
calculate notification time
Send notifcation
Student finishes task
Mark as complete
Paste status

The activity diagram consists of 2 web pages. The student web page and the web server. Thus we have 2 swimlanes.
The student enters his webpage and starts to insert his tasks. The student will specify the content and the deadline for each task. Then the web server will save the tasks into the tasks table in the database. After that the server will notify the student according to the set preference of the notification. Finally, when the student completes the task it marks it as completed and the program terminated.

c. Sequence Diagram



Student GUI
Web Server
database
Student
enterToDoListPage()
enterToDoListPage()
page entered
display the to do list page
specifyContent(), specifyDeadline()
sendContent(), sendDeadline()
saveContent(), saveDeadline()
content and deadline saved
createNotification()
selectNotificationPreferences()
notification preferences selected
readNotificationPreferences()
saveNotificationPreferences()
sendNotification()
notification preferences saved
Notification sent
markAsComplete()
markAsComplete()
updateStatus()
Status Updated

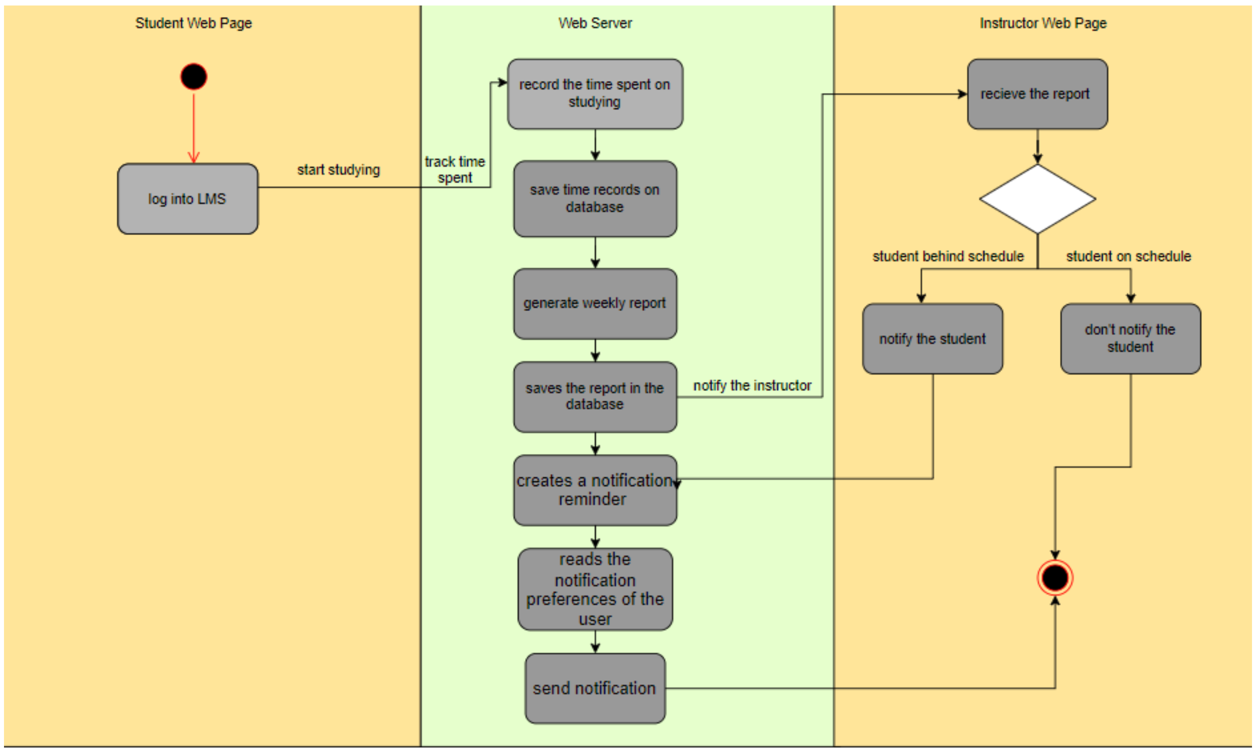
The classes are: Student GUI, Web Server(NoteHandler), and Database.
The actor involved: Student
To begin with, the student will first enter the to-do list page by pressing on a button and in its turn the Student GUI will then call the method enterToDoListPage() in the Web Server. So the GUI class will be an intermediary between the Student and Web server. After the server enters the to-do list page it will return a message "page entered" to the GUI class and then the GUI class to the student with a return message "display the to-do list page". Afterwards the student will specify the content and the deadline through the Student GUI class (specifyContent() and specifyDeadline()). The GUI will send the content and deadline to the Web Server by calling sendContent() and sendDeadline() methods. Finally the content and deadline will be saved when the Web Server calls saveContent() and saveDeadline() in the Database class. The Database class will send a return message to the Web server "content and deadline saved". Once the student is done with the content and deadline, the student will then select the notification

preferences and then the Student class will return a message to the GUI class (“notification preferences selected”). The GUI class will call the method readNotificationPreferences() in order to make the Web Server read the student’s preference and will in its turn ask the Database to save the preference by calling saveNotificationPreferences(). The Database will send a return message informing the Web server that the notification preferences are saved and then the Web server will send a notification to the GUI class using the sendNotification() method. Thus, the GUI will display the notification to the Student. Once the student finishes his task he will mark it as complete in the GUI Class and then the GUI class will call the method markAsComplete() in the Web Server. At the end the server will send the status of the task to be updated in the database by calling updateStatus() and then finally the database will send a return message that the status is updated.

D. Track Study Hours Diagram:
    a. Use case Scenario

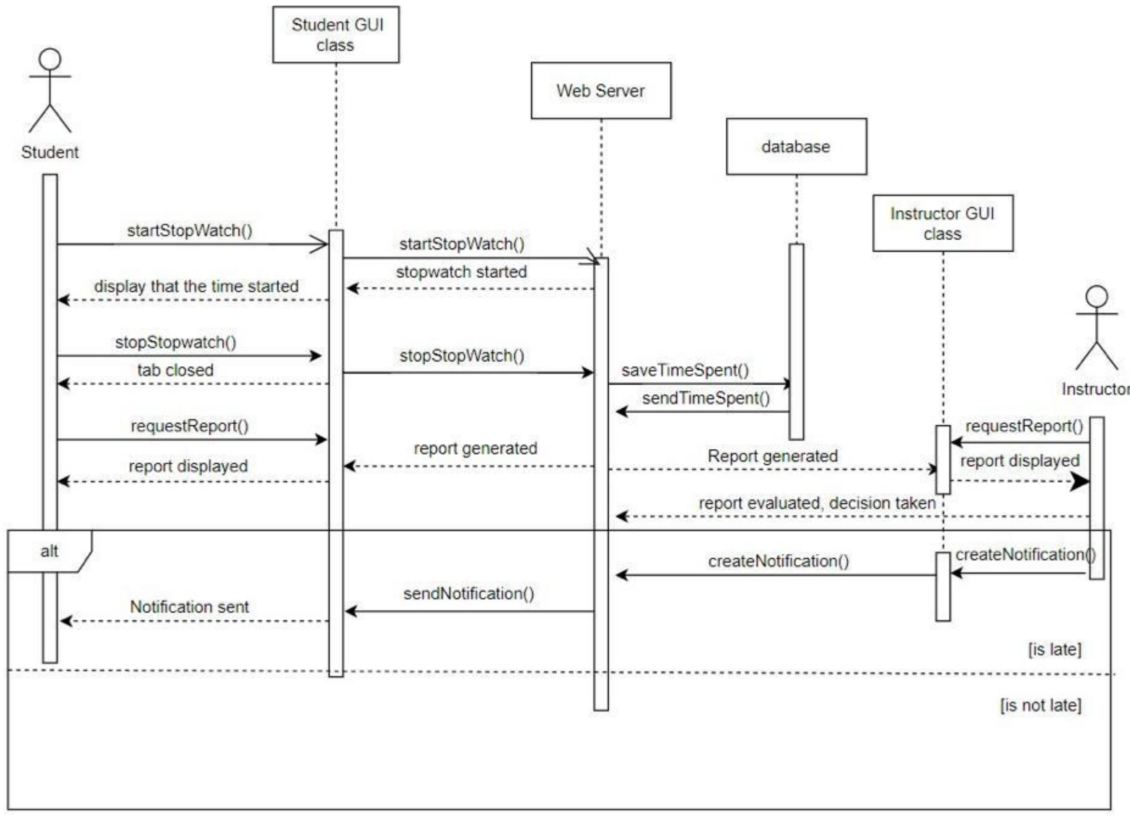
Use case name: Evaluate study Hours	
Area: All the Web pages	
Actors: Instructor and student	
Description: The time that the student spends using the VLA will be recorded by the system and sent to the instructor. Then the instructor will evaluate these hours and send a notification to the student if needed.	
Stakeholder: Students and instructors	
Level: Green	
Triggering Event: The student open the VLA	
Trigger Type: **[✓ External]** **Temporal**	
Steps Performed	**Information for Steps**
1. The **student** logs into the **VLA** and *starts studying*	The student username and password
2. **The server** *records* the time that the students spends studying using **stopwatch**. The time will start when the student *opens the VLA* and stopped when the *student closes it.*	The student's study hours
3. When the student closes the tab of the VLA, the **web server** *sends the time spent* and will added it to the **timeSpent table** in the **database.**	
4. Every Sunday, the system *generates* a **report** including **name and id of the student, name of the instructor, name of the course and number of hours that the student spent on each chapter** and *compare it* to the suggested number of hours for that chapter.	The calculations the system needs to do to generate the report
5. The server saves the report to the database and send it as a **notification** to the instructor.	
6. The instructor *evaluates the report* of each student and *decide* whether he should *notify the student* or not.	The report generated by the system
7. If the instructor sees that the student is behind schedule he can notify the Student using the LMS notification feature.	Student study hour recorded in the report
8. The systems *creates a notification reminder* in the TimeSpent table in the database. The notification reminder contains the **deadline of the task**.	List of the students that the LMS should notify
9. While the system is running, it *reads the data* from the timeSpent database every 1 minute.	Data from the timeSpent table
10. If the notification time is now, the system *sends a notification* to the student by *calling the notification web service* in the LMS	
Preconditions: The student is logged into the LMS and started using the VLA functionalities	
Postconditions: The study hour is recorded and the report is generated	
Assumptions: The student is using the VLA to study	
Success Guarantee: The instructor receives a report on the study progress of each student and can notify the students that he wants.	
Minimum Guarantee: The study hours are recorded and a report is generated by the system	
Objectives Met: The instructor can spot the students behind schedule and notify them to keep them on track.	
Outstanding Issues: What happens when the student opens the VLA but left his device?	

b. Activity Diagram



The activity diagram consists of 3 web pages. The student webpage, web server and instructor web page. Thus we have 3 swimlanes.
After the student logs in to the VLA, the time starts to be recorded. While the student continues studying, the web server records the time , saves data to the database,generates a weekly report and saves the report to the database. After the report is saved in the database, the server will notify the instructor and send it the report. When the instructor rcvs the report, it will check it, if the student is behind the schedule, he will notify the student and the program will be terminated. Otherwise, no notification happened and the program terminated.

c. Sequence diagram

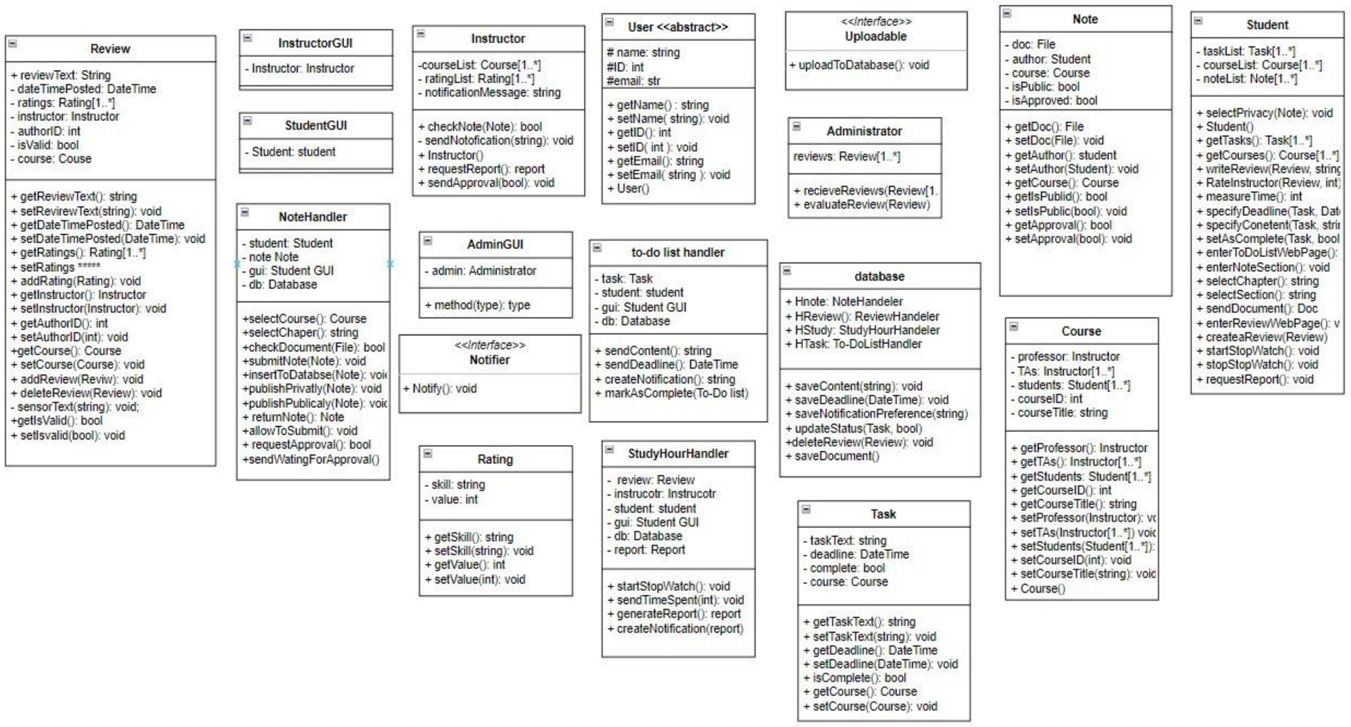


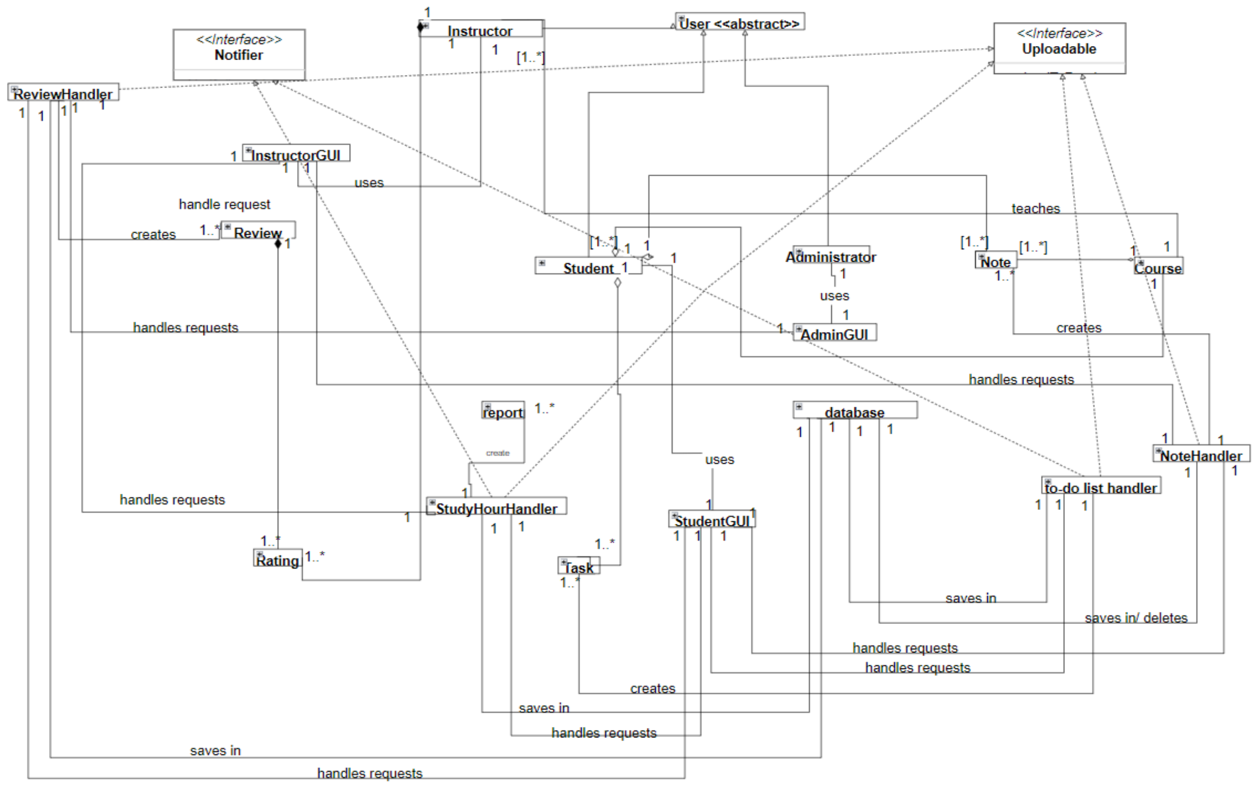
The classes that are involved are: Student, Student GUI, Web Server(StudyHourHandler), Database, Instructor GUI and Instructor
To begin with, when the student wants to track his study hours, he will start the stopwatch present in the homepage of the web page, the study hour will keep counting until the student stops it. These two actions would be done by the student using the startStudyHours() and stopStudyHours() respectively. When the student starts the stop watch he will be interacting with the Student GUI, The student GUI will be sending these methods to the web server who in his turn will return the messages "stopwatch started" and "tab closed". When the student stops the stop watch or closes the tab, the gui will send the info to the web server the time that the student spent studying, the server will send this data to the database to save it there. The database will save it using the saveStudyHours() method. Sunday night when the server needs to generate the student report, The database will send to the

database the total study hours of the student using the sendStudyHours() method and the server will generate a report on each student. When either the instructor or the student asks for the report, they will do it through the method requestReport(), the server will send the report to their respective GUI who will display the report to the user. When the instructor views the report, they will evaluate it to check if the student is late on his work or not and he will send this decision to the web server. If the student is late, the web server will create a notification using the createNotification() method and will send notification to the student's GUI using the sendNotification() method.

V- VLA Structure

A. Class diagram: (The first photo is to show the classes only and the second photo is to show the relationships between them. We did that just for cleanliness and clarity)





According to the sequence diagrams and objectives of the VLA we develop the above class diagram.

To organize the class of this system into packages, we can first make a package named “users” that contains the student, instructor, administrator and user classes. Then, we would create a package for each one of our objectives: first we have a package to-do list that contains the task, task handler classes. Then the note package that contains the noteHandler and note classes. Then we have a review package that contains the review and reviewHandler and Rating classes. The last package of this category is the study hours package that contains the report, StudyHourHandler. We would also have a Web package that contains the Student and instructor and administrator GUIs and then we would have a SQL package that contains the database and finally we would have the course class which can be put in a package alone.

The student, admin, and instructor inherits from abstract class users because they share common variables and methods such as the name, id , email and others.

The Uploadable interface contains only the UploadToDatabase() method , and each class implementing this interface should have this method.

Note handler, to-do list handler, Review handler and study hours handler are classes that are related to the system functionality. They all have the same method which is uploading to a database. Thus all the 4 classes implement the uploadable interface.

If the student was late on his tasks or behind its schedule, the VLA notified the student. Thus the classes to-do list handler and study hours handler share the same method which is to notify the student. Then they implement the interface notifier which has a notifyStudent() method.
The classes that implement this interface must implement notifyStudent() method.

The Instructor has an instance of rating class. The existence of rating depends on the existence of the instructor. Thus the relationship between the 2 classes is Strong relationship (Composition). The relation between the instructor class and rating class is 1 to * (many) since the instructor can have many ratings.
Moreover, the instructor “uses” the instructor GUI. Thus, there is a 1 to 1 association relationship between the Instructor class and Instructor GUI class.

The review handler will always “create reviews” based on some requests , thus there is an association relationship between review class and review handler class. Similarly for the note and task classes and their system handlers. The relation between the handlers and these classes is 1 to many since the server will issue many requests to make new reviews/ notes, tasks.
The review has an instance of rating class. The existence of rating depends on the existence of the review. Thus the relationship between the 2 classes is Strong relationship (Composition). The relation between the review class and rating class is 1 to (M) since the review can have many ratings.

The task is created through the task handler thus there is a relationship “creates” between the todolist handler class and Task class (1 to many).

As stated in the Student class, the todo list Handler handles the tasks that the students add. Thus we have the “handles requests” relationship between the todo list handler and the Student GUI.
When the tasks are added , the tasks are saved in the database through the todo list Handler thus the relationship “saves in” exists between Database and todo list Handler.

The review is created through the review handler thus there is a relationship “creates” between the review Handler class and review class (1 to many).
As stated in the Student and Admin class, the review Handler handles the reviews that the students add and the admin checks. Thus we have the “handles requests” relationship between the review handler and the Student and admin GUI.
When the reviews are added, the reviews are saved in the database through the review Handler thus the relationship “saves in/deletes” between Database and review Handler.

The instructor GUI class acts as an intermediary interface between the instructor and other functionalities, Thus there exists a “handle requests” association relationship (1 to 1) between the instructor GUI class and other system handlers such as “studyhourhandler” class and “note handler” class.

The instructor “teaches” 1 or many courses and the course can be taught by one or many instructors. Thus we have 1..* to 1..* association relationship between Instructor class and Course class.

The database class has association relationships (1 to 1) with the system (or server side) handlers (Note handler, to-do list handler, review handler and study hours handler). This is necessary because the web server always issues queries to the database.

All the functionality handlers classes (Note handler, to-do list handler, review handler and study hours handler) acts as web server side in which they communicate with the users GUIs and the database.

The student class contains a list of tasks, notes and courses. The existence of tasks and notes depend on the existence of the student. Thus it is a strong relationship (Composition). On the other hand, the existence of course does not depend on the existence of the student (weak relationship). Thus the relation between student class and course is Aggregation.
In addition, the student GUI has an instance of the student class therefore the student is related with the student GUI in a “uses” relationship. The review handler, note handler, study hours handler and the to-do list handler all have an instance of the class student in them so there is a “handles request” relationship between the student GUI class and each of them.

The students in the Notes section have to choose the course between the list of courses that he/she wants to upload the notes to. Each course has its own notes, so the cardinality is 1 to [1..*] Therefore, we have the “has a” relationship between Note and Course classes.

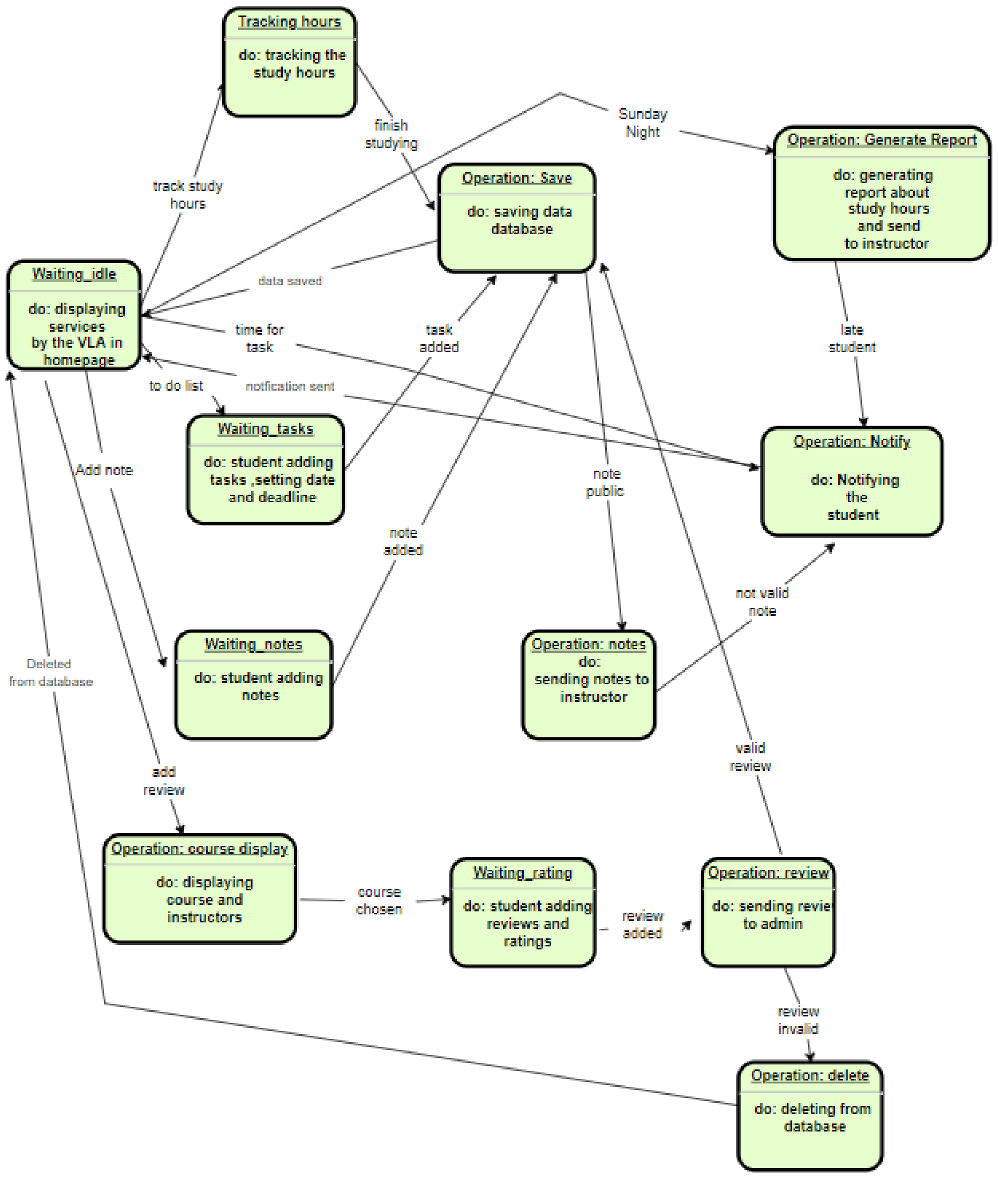
For the admin to be able to carry out his operations he/she will use Admin GUI that will act as an intermediary and thus the “uses” relationship between Admin GUI and Admin class.
The review handler will handle the review taken from the admin GUI class in case it was inappropriate or appropriate and thus the relationship “handles requests” (1 to 1).

The note is created through the note handler thus there is a relationship “creates” between the Note Handler class and Note class (1 to many).
As stated in the Student and Instructor class, the Note Handler handles the notes that the students upload and the instructor checks. Thus we have the “handles requests” relationship between the Note handler and the Student and Instructor GUI.
When the notes are uploaded whether it is public or private, the notes are saved in the database through the Note Handler thus the relationship “saves in/deletes” between Database and Note Handler.

The report is created through the studyhours handler thus there is a relationship “creates” between the studyhour Handler class and report class (1 to M).

As stated in the Student and Instructor class, the studyhour Handler handles the report that provides data about the time that the student spent on studying and the instructor checks. Thus we have the “handles requests” relationship between the studyhour handler and the Student and Instructor GUI.
When the student finishes studying and the report is generated, the report is saved in the database through the studyhour Handler thus the relationship “saves in” between Database and studyhour Handler.

B. State Diagram:



Tracking hours
do: tracking the study hours
Operation: Generate Report
do: generating report about study hours and send to instructor
Operation: Save
do: saving data database
Waiting_idle
do: displaying services by the VLA in homepage
Waiting_tasks
do: student adding tasks ,setting date and deadline
Operation: Notify
do: Notifying the student
Waiting_notes
do: student adding notes
Operation: notes
do: sending notes to instructor
Operation: course display
do: displaying course and instructors
Waiting_rating
do: student adding reviews and ratings
Operation: review
do: sending revie to admin
Operation: delete
do: deleting from database
finish studying
Sunday Night
track study hours
data saved
time for task
task added
late student
to do list
notification sent
Add note
note public
note added
not valid note
Deleted from database
valid review
add review
course chosen
review added
review invalid

State	Description
Waiting idle	The system is displaying the homepage and listing the services of the VLA.
Waiting tasks	The system is waiting for the student to enter the tasks ,their date and time in the to do list page.
Waiting notes	The system is waiting for the student to upload the notes , and choose the visibility to be private or public.
Tracking hours	The system is measuring the hours that the student spent on each chapter
Operation: Save	The system is saving the data (notes, tasks, reviews, hours) to the database.
Operation: Course Display	The system is listing the courses that the student enrolled in.
Waiting rating	The system is waiting for the student to put his input in the reviews and rating sections
Operation: Delete	The system deletes a row from the database when needed
Operation: Review	The system sends the review made by the student to the administrator to check if the review is appropriate
Operation: notes	The system sends the notes uploaded by the student to the instructor to give his approval
Operation: Notify	The system notifies the user when necessary
Operation: Generate Report	The system generates a report based on the study hours tracked of each student to send it later on to the instructor

Stimulus	**Description**
Track study hours	The student enters the VLA and starts studying.
Finish studying	The student finishes studying and measuring time stops.
Sunday night	The date is sunday. Each Sunday the VLA will generate a report about the student performance.
Late student	The student is behind his schedule
Not valid notes	The notes contains invalid data (contain errors)
Notes public	The student chooses to make his notes public
Notes added	The student finish uploading the notes
Add review	The student chooses to add review on a certain instructor
To-do list	The student enters the to do list web page
Time for task	The deadline for a certain task is reached
Data saved	The data is saved to the database
Task added	Tasks are added to the database
Valid reviews	The reviews are valid and do not contain impolite words.
Course chosen	The student chooses the course that he/ she wants to evaluate its instructor.
Deleted from database	An inappropriate review is deleted from the database
Review added	The review in approved and is published
Review invalid	The review submitted by the student was inappropriate
Add review	The student filled the review input on clicked on the submit button

VI- Conclusion:

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. This will help the software engineers as well as the customers for better understanding of the system before building it or before having a real view. Moreover, all the models implemented in this phase will represent different photos or perspectives of the VLA system. This will facilitate the role of software developers in the implementation phase. For example, the context diagram helps in the implementation by showing what functionalities of other systems should be integrated with our system. The class diagram will help the developers in discovering the classes that they will use when developing the software and the associations between these classes will help in identifying what relationship between OOP relationships that the software developers will use in the classes implementation. Activity and sequence diagrams also help in the implementation phase by helping the developers to know the sequence of calling the methods. Use case diagram and architecture diagram helps in identifying the interaction between the system classes and its environment. Similarly, a state diagram helps in identifying the current state of the system and may be the classes when implementing the system. Thus system modeling or system design is an essential step in software engineering.