LAU
الجامعة اللبنانية الأميركية
Lebanese American University

SPRING 2022

Software Design and Architecture Document

for

LAU Virtual Learning Assistant

Based on LAU VLA SRS Version 1.0

Prepared by Osama Shamout, Omar Shatila, Omar Mlaeb

April 15, 2022

Document Reference Number: OPENG/D&A-LAU-VLA-V1.0

Copyright Notice

Founded in 2022, Op-Engineers Ltd., had its humble beginnings in the heart of Beirut at the Lebanese American University campus. The company has been an extraordinary technological service company providing innovative technological solutions in the fields of Software Engineering and Software Development. By following the ISO/IEC/ STANDARD IEEE 29148, Op-Engineers Ltd. was able to establish its presence by providing high-quality products that fulfill the market needs.

Proudly Engineered in Lebanon.

This page intentionally left blank.

Table of Contents

1. Introduction

The purpose of the this Design and Architecture document is to showcase our VLA system designs. We start with the Context Diagram to present a visual reference to stakeholders and engineers about the VLA System's environment (external actors) and their interaction with the system. Then, we applied the Architecture Diagram to visualize the components of the VLA System in which we divided diagram into several subsystems that encompass the VLA's goals. After drawing the Architecture Diagram, we were able to deduce the Use-Case Diagrams with reference to the SRS document previously provided, forming the first step in our Interaction Model. This step was pivotal in understanding the actors' interactions with the system as it helped to clearly set the user requirements as functional "goals" in the Use-Case diagram. Also, it shall provide a clear and easy-to-read diagram for stakeholders to view the expected system's functionalities. After that, each member of the team chose a Use-Case to be extended for a Use-Case Scenario, an Activity Diagram, and a Sequence Diagram. Thus, the Sequence Diagram provided a fuller picture by clarifying more details for the Behavioral Model in conjunction to the Interaction Model, displaying the system's inputs and output, and the communication paths/interactions between the different system objects. In general, the Behavioral Model was useful in depicting the dynamic behavior of the system through the activity and the sequence diagrams. In other words, the system's response to stimuli from its environment. Also, the Activity Diagram was useful in showing the flow logic of the Use-Cases shown previously.

After forming the Interaction Models and Behavioral Models, we got the necessary documents to create the Structural Model. The Structural Model utilizes the UML Class Diagram in depicting the different classes and the relationships between them. This was possible through analyzing the Use-Case Scenarios and identifying the keywords and listing them as objects and methods. This step has been essential in correctly modeling the UML Class Diagram as we take our objects and method from the real-world scenario. After creating the classes, we added the relationships and the multiplicity values formed. Finally, to complete our models, we supplemented the State Diagram, to portray the different states the system moves to as it transitions due to internal or external events.

2. VLA Context and Architecture

2.1 Context Diagram

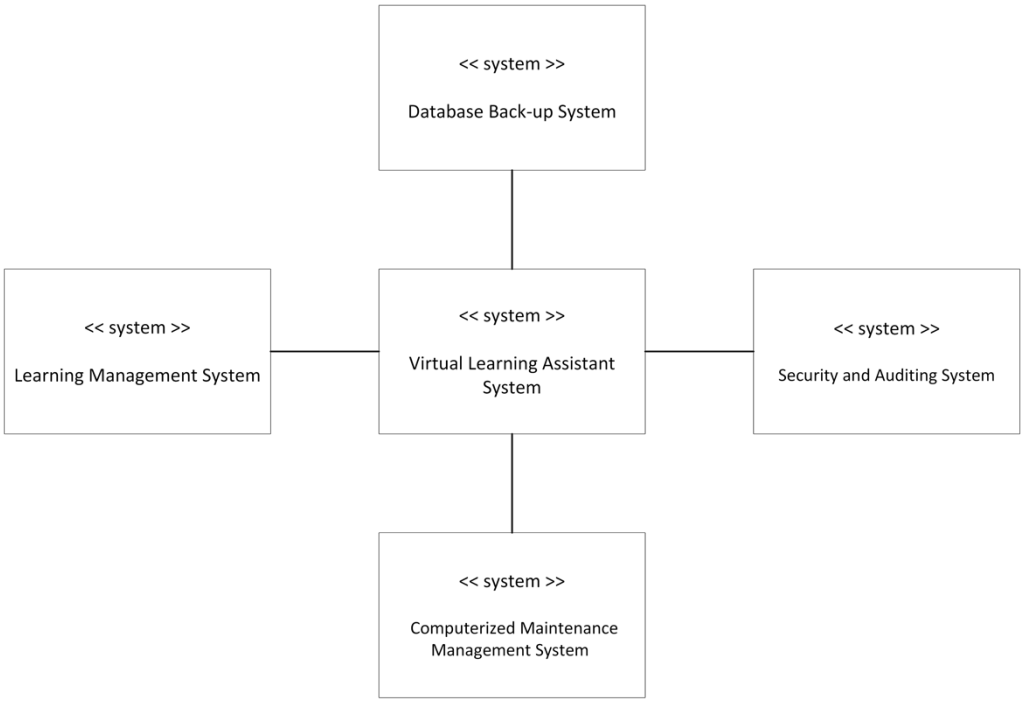


Figure 1: Context Diagram

The above Context Diagram depicts the VLA System and the external systems interacting with it. The VLA System interacts with four external systems. The Learning Management System exchanging data to provide for the system the functionalities to run the Task Management, Content Query, etc. The Data Back-Up System interacts with the VLA by backing it up manually or automatically as specified by the SRS. The Security and Auditing System which is run by the administrator to scan and analyze the system to ensure the SCAP security standard. Finally, the Computerized Maintenance Management System tracks the scheduled maintenance activity, generates system insight, and tweaks performance.

2.2 Architecture Diagram

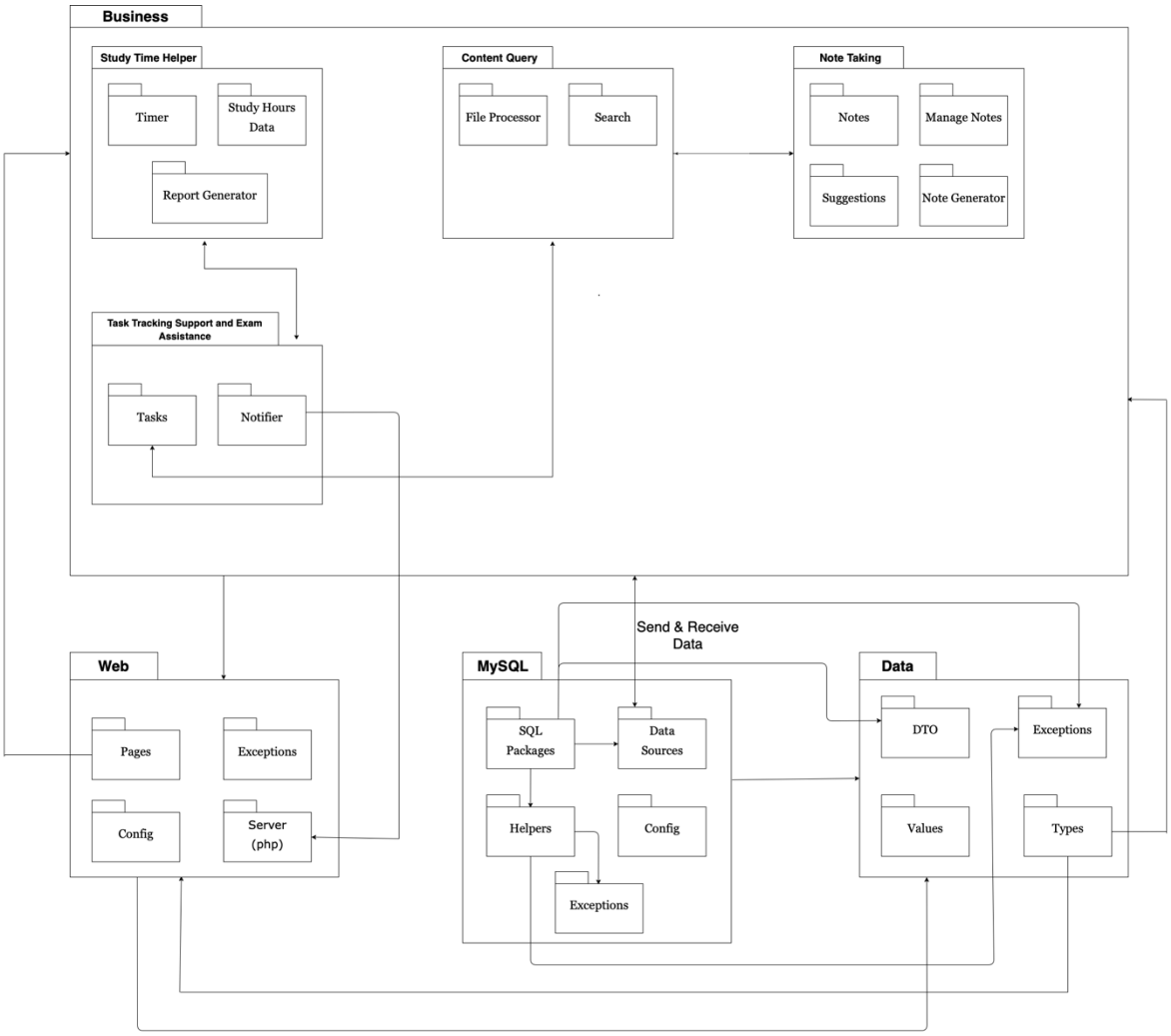


Figure 2: Architecture Diagram

The Architecture Diagram of the VLA system is composed of the main VLA System inside the system component. Inside that component are four different components. Each representing a subsystem. The Study Time Helper package is a subsystem that contains the interactions that generate the timer, the study data, and the reports resulting from the latter. The Study Time Helper interacts heavily with the Task Tracking Support and Exam Assistance. These two subsystems (deduced from the objectives) are combined as their functionalities are similar and hence it would reduce the processing time by making these two subsystems in the same component. The notifier is also present in this package which is triggered upon certain a event occurring (external or temporal) and hence it sends the request to the Web Server to

process that request. The Task component is also a part of the aforementioned subsystem, and it contains all the Task Creation and Management details. It interacts with the Content Query as student might search for a certain task. The Content Query subsystem is responsible for managing the files and reading them as well as the search functionality. This subsystem renders files from the database to be read by the VLA and hence “searchable” via the Search component. Hence, it is pivotal in its role in providing the material a reading mechanism to process the files in the VLA system. Next, the search component searches for material across the internet and the VLA System to return the results for the user, accomplishing the Content Query objective. Finally, the Content Query component also interacts with the Note Taking component, the last subsystem in the system. The Note Taking component is responsible for the Notes Management (creating, deleting, updating, etc.). Coupled with the Content Query Component, the Note Taking component can enable student to search through notes and process the notes files using the File Processor to produce the Notes as readable PDFs. Finally, the Database component is connected to send and receive data to the whole VLA System. The Data Types also communicates to the VLA System. The webpages of the Web Server also interact with the VLA System, while the VLA System sends information to the Web Server. The Web Server, also interacts with the Data saved.

3. VLA Interactions

3.1 Use-Case Diagram

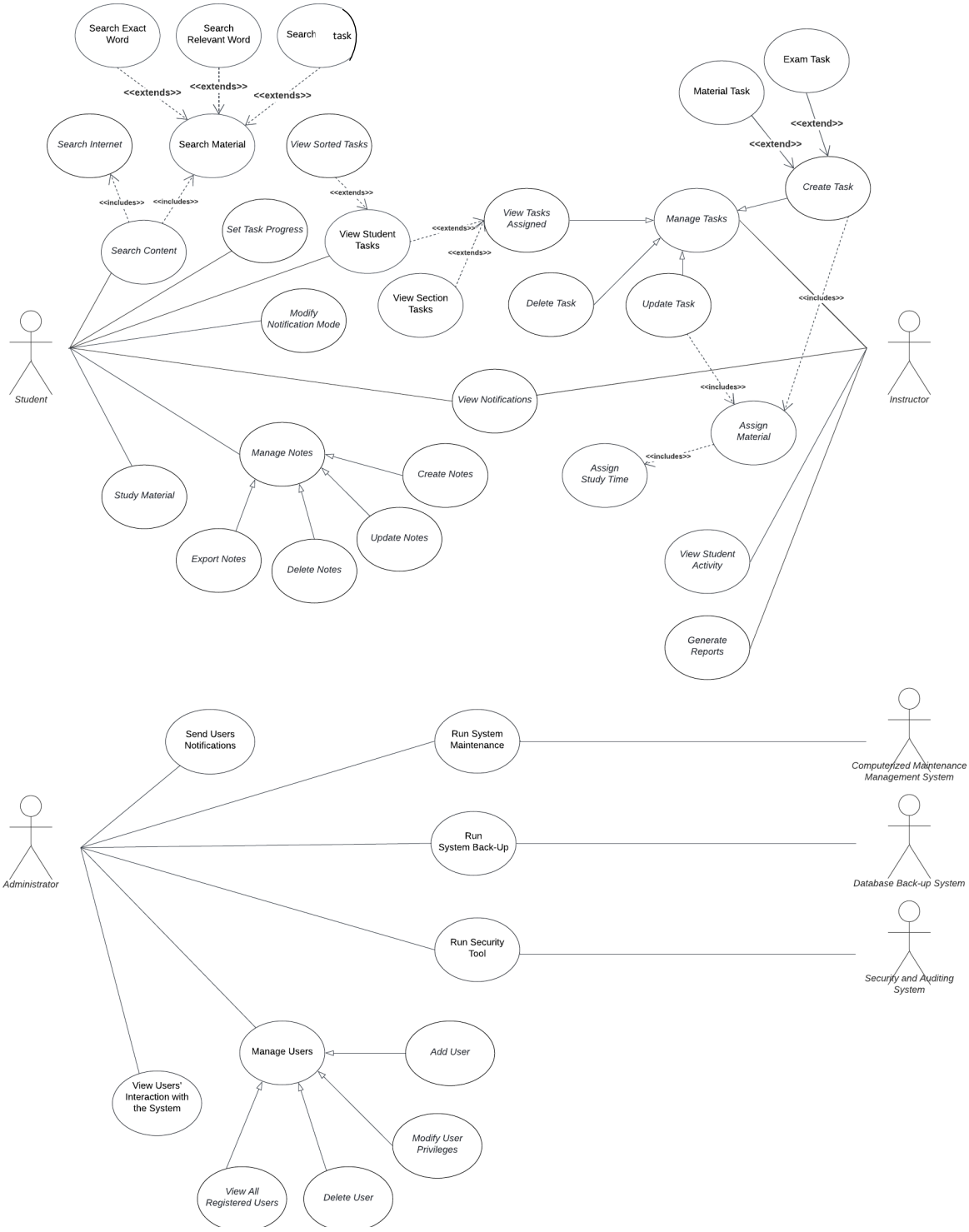


Figure 3: Use-Case Diagram

First, we identified the actors that will be using the system from the SRS document. Afterwards, we identified the subsystems that will require an interaction between the actors and the system. Hence, we created the Manage Tasks to encompass most of the Task Tracking Support objective. It clearly shows that the “Instructor” actor is the major user of these functionalities. Further, Manage Tasks is connected partly with the Study Time Helper which should supplement the functionality of tracking the students and providing feedback to the student and the instructor about the study times compared to their peers in the section. Moreover, we included the Manage Notes to take care of the Note Taking objective for students which mostly revolves around goals that are centered around the “Student” actor. The Manage Notes includes everything that a student might need from creating, modifying, deleting, or exporting notes (which could be sent to the Instructor or to the device the Student is using). The next major objective was Search Material Use-Case which encompassed a wide variety of search functions that are a generalization of Search Content. The search features enable the student to search for material by exact word, or by relevant word (synonym to the word input). Moreover, the search feature will query the internet and display the top three most relevant results which shall help the student in supplementing their studies with outside resources. Finally, the Use-Case encompasses the most major Use-Case of an “Administrator” actor which revolve around backing up, maintenance, managing users (add, delete, update, and/or view users), and security. The “Administrator” can also view the logs generated and send notifications to users. This is essential, as when an “Administrator” is going to create any of the maintenance, security patches, etc., they need to notify the users about the changes that shall occur and if any disturbance to service shall happen. Additional functionalities to the “Student” are set track progress to override the task progress manually. The view tasks list, which could be sorted by importance (eg. exams then material tasks). Also, the Student can modify the trigger to the notification by the Modify Notification Mode. The Instructor’s additional functionalities are also viewing the tasks assigned, and the student’s progress, generate reports. Both student and instructor share the view notification Use-Case as the functionality is the same on the VLA System.

4. VLA Components

4.1 Use-Case Scenario 1

Use Case: Create Task

Use case name: Create Task	
Area: Task Management	
Actors: Instructor	
Description: This use case allows any registered instructor in the university to create tasks in the system in order to be assigned for students enrolled in their section.	
Stakeholder: Instructor, Student, Administrator.	
Level: Blue	
Triggering Event: An instructor enters Manage Tasks menu clicks on Create Task on.	
Trigger Type: ■ **External** □ **Temporal**	
Steps Performed (Main Path)	**Information for Steps**
1. System receives Create Task request.	Create Task Request.
2. System fetches to Web Server instructor information from LMS.	Instructor's LMS Database Records.
3. System send Create Task Webpage to instructor.	Create Task webpage.
4. Instructor Chooses Section(s) (if more than one) to apply task for.	Instructor's active sections.
5. Instructor specifies task type.	Create Task webpage.
6. Instructor assigns material(s) for task.	LMS uploaded material by instructor
7. Instructor specifies time to finish task.	Instructor's estimation time for task completion.
8. Instructor clicks on Confirm.	Create Task webpage.
9. System registers task.	Instructor Task Record.
10. System adds/assigns task for students.	Enrolled Students in Section Record.

	Students Tasks Record.
11. System sends notification to student about the new task.	Send Notification API.
12. System sends notification to instructor about the new task.	Send Notification API.
13. System displays confirmation on create task webpage.	Create Task webpage.
Preconditions: The instructor has been registered in the system. The instructor has logged in the system. The instructor has students enrolled in their section.	
Postconditions: The instructor has successfully assigned a task to their section. The system has sent notifications to instructor and student about the new task.	
Assumptions: The instructor knows how to navigate to homepage to see Manage Tasks menu. The instructor knows how to navigate to Create Task from the Manage Tasks menu. The instructor has privileges to create tasks. Browser supports JavaScript. The instructor allows cookies.	
Success Guarantee: The instructor has successfully assigned a task to their section.	
Minimum Guarantee: The instructor was able to view the create task webpage.	
Objectives Met: The system shall enable instructors to assign tasks to students.	
Outstanding Issues: How to retrieve uploaded material from LMS to VLA quickly. How to create APIs to communicate between VLA and LMS.	
Priority (optional): High	
Risk (optional): Medium	
Exceptions: The instructor has input time to finish task less than 0 which cancels the add task and	

Figure 4: Use-Case Scenario

4.2 Use-Case 1's Activity Diagram

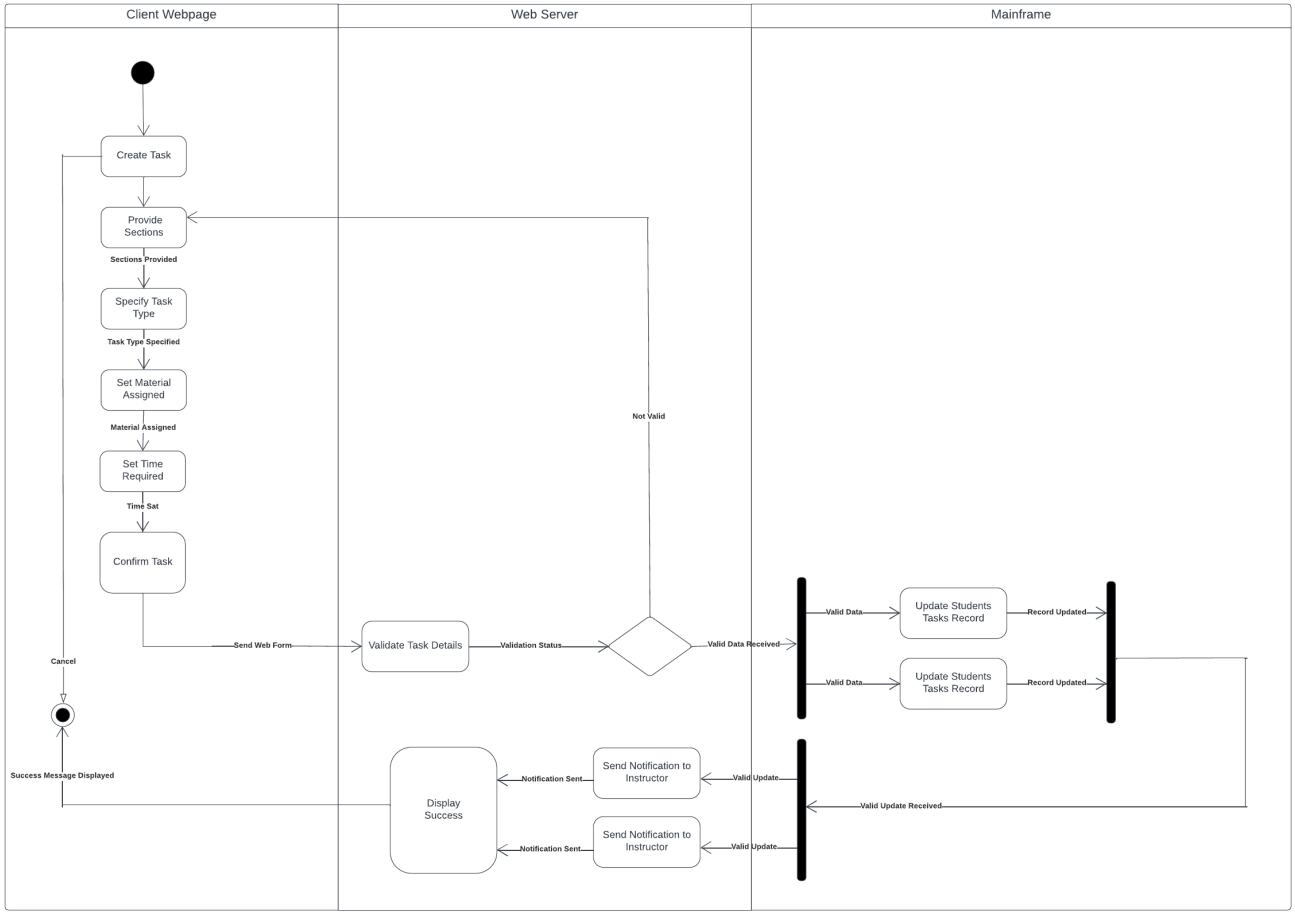


Figure 5: Activity Diagram

The system starts by the instructor's click on Create Task from the Manage Tasks menu available on the Homepage. From there a form shall be displayed to be filled by the instructor. The form provides the information of the section as a dropdown menu from the LMS Database, then the instructor inputs the task type (Exam or Material), then assigns available uploaded material from the LMS Database for the assignment. Afterwards, the instructor shall provide the time required to complete this task. Then the instructor clicks on confirm to proceed with the Use-Case. From there, the VLA Server verifies if the information is valid by checking if the study hours are above 0. If not, the system returns the command to fill the section field in the form. If the information is valid, the system sends the information from the Server into the VLA Database, the VLA Database shall update the student and the instructor's records. Adding the task assigned to each user category respectively. The system waits for this step to be done in

synchronization to go to the next step. After the update has been complete. The flow goes back to the VLA Server which triggers a send notification in synchronization to both the instructor (informing them about the new task created), and the student (informing them about the new task assigned to them). The server generates a successful message and inputs it to the webpage to be displayed to the instructor. Then the use-case terminates.

4.3 Use-Case 1's Sequence Diagram

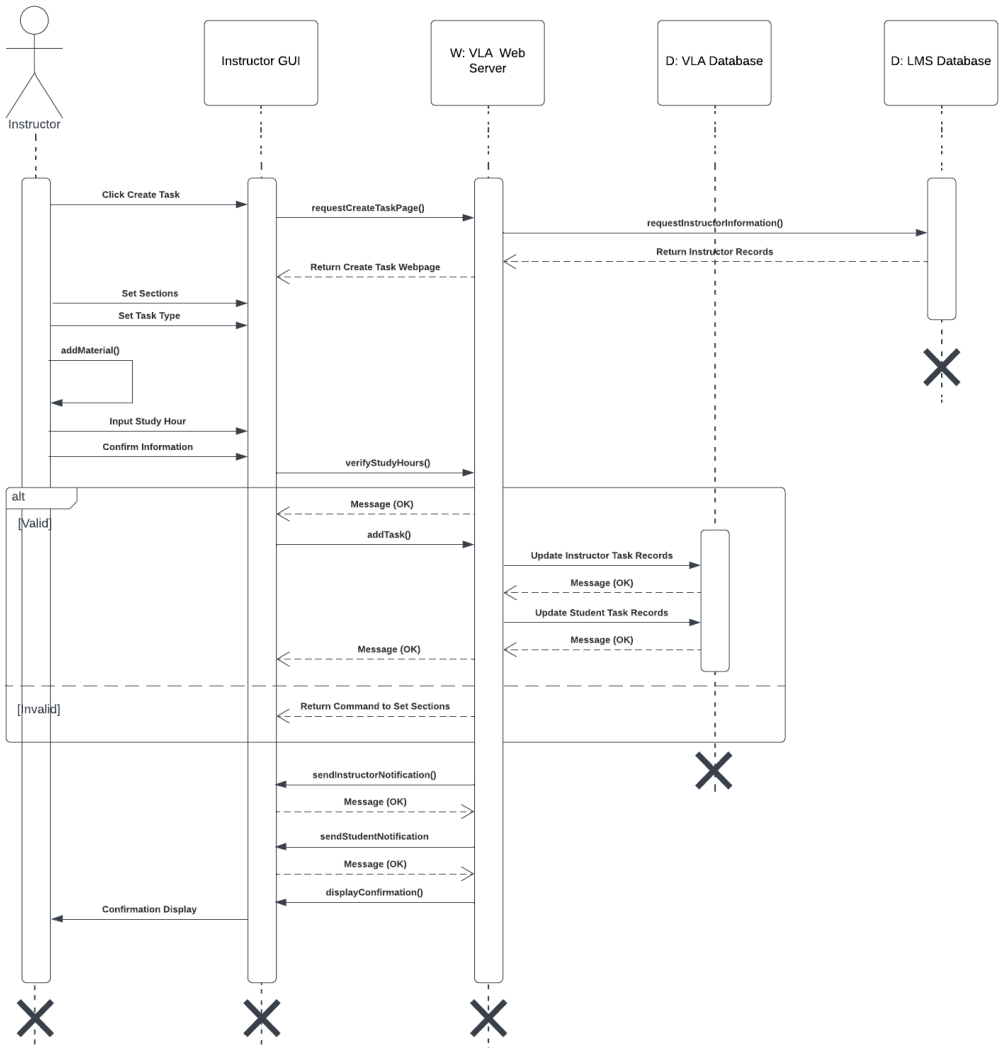


Figure 6: Sequence Diagram

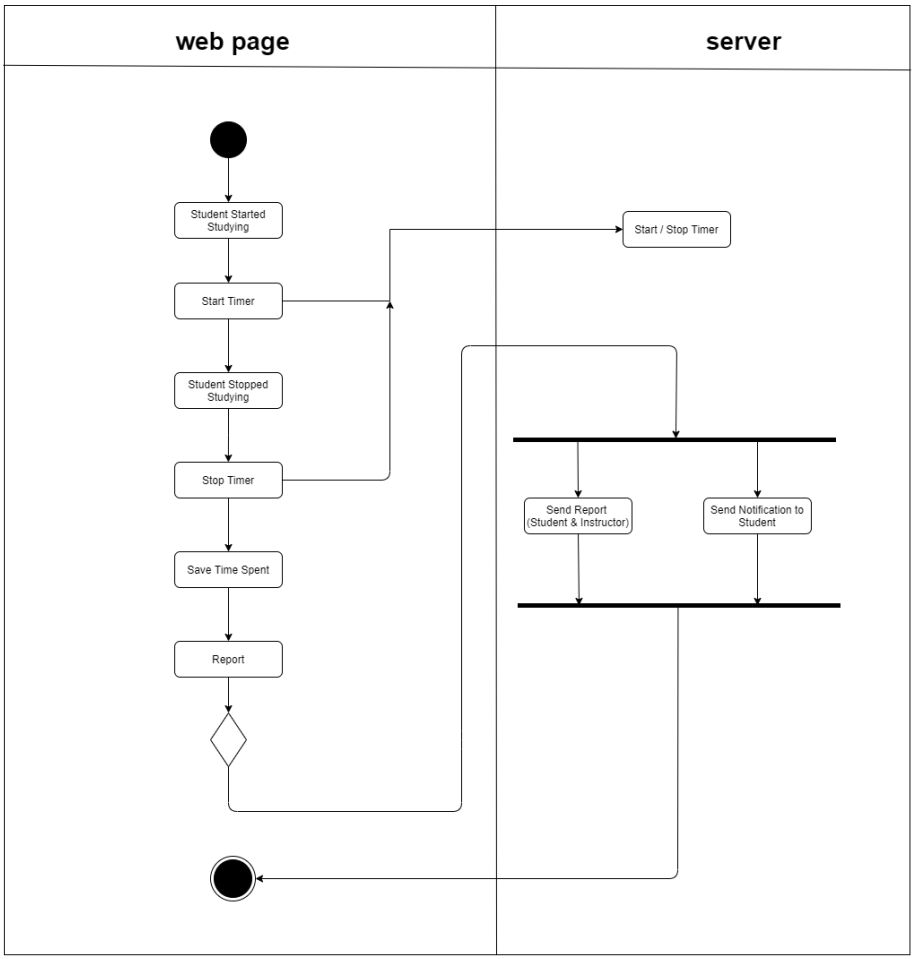
Similar to the explanation of the Activity Diagram. The Use-Case gets triggered by the user's click on Create Task button. The button triggers a request to the web server to fetch the Create Task Webpage. The Create Task Webpage requires information to be fetched from the LMS DB (Instructor's section(s), and

material(s)). Hence, a request from the VLA Server will be issued to the LMS DB that shall return the section(s) and material(s) uploaded to the LMS. After, the successful return, the system displays a Create Task Form which enables the instructor to graphically fill in the information for the task to be added. The instructor from there chooses from the drop-down menu the section they want to assign the task to, then, the instructor indicates the task type: Exam or Material. Afterwards, the Instructor adds material from the drop-down menu for add material. Further, the add material is a button to be clicked which shall regenerate another add material drop-down. Hence, the instructor can assign as many materials for the task as they want. Then the instructor inputs the required study hours for the task to be assigned. Then the Instructor clicks on confirm, sending the input study hours to the server for verification on the VLA server. If the system found the invalid information (the study hours are less than 0) the system returns the user to the set section to input a section (in essence, refilling the form). If valid, the VLA system sends the input information by triggering the addTask() method. This method sends an update to the VLA DB to update the records concerning the tasks for the instructor and the student to reflect the addition of a new task. From this successful return requests returned from VLA DB to VLA Server, the VLA triggers a sendNotification() method for both the instructor and the student to notify them about the new added task (The notification is visible via a notification button that displays all notification for the user). After sending the notifications, the system generates a successful message on the server which is then sent to the Create Task webpage to be displayed to the user indicating the success of their Task Creation.

4.4 Use-Case Scenario 2

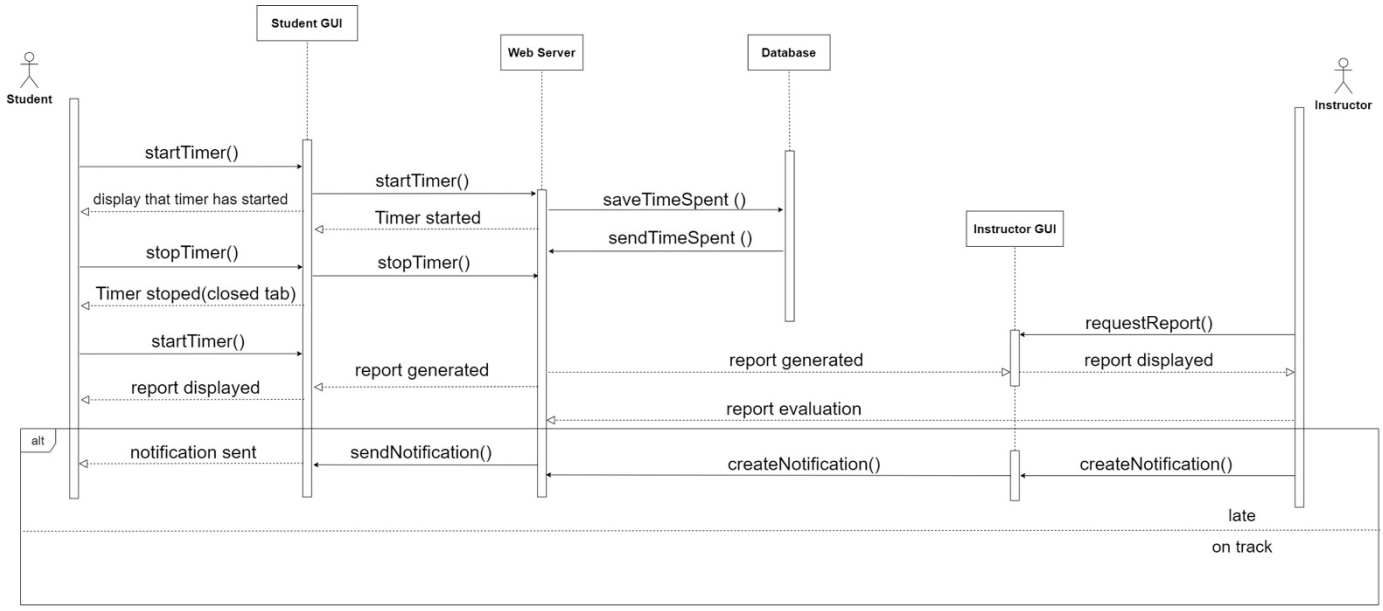
Actors: Instructor and Student	
Description: This use case allows the instructor to track the studying hours for each course of the students and if they are reaching the recommended studying hours for their course/section.	
Stakeholder: Instructor and Student	
Level: Blue	
Triggering Event: the student or the instructor can see the studying hours once the student begins studying the material	
Trigger Type: ■ **External** □ **Temporal**	
Steps Performed	**Information for Steps**
1. System starts the timer when the student start	Start Timer
2. Sends time started from webpage to student	Timer webpage
3. System stops time after students finishes studying	Stop Timer
4. System saves the time when the student stopped sent from webpage to the database	Timer Data
5. Send from database the spent time to webpage	Timer Data
6. Generates report webpage to users	Generates Report
7. Send notification to student	Display Report
8. Send notification to student	
Preconditions: student or instructor should have an account and registered in the course	
Postconditions: The student is behind the material or not	
Assumptions: The user must be a student or instructor. The user knows the URL of the system website	
Success Guarantee: user can know how mush time he spent on studying	
Minimum Guarantee: student was able to study and finish some tasks and the instructor was able to track study hours of the student	
Objectives Met: track the study hours of the student	

4.5 Use-Case 2's: Activity Diagram

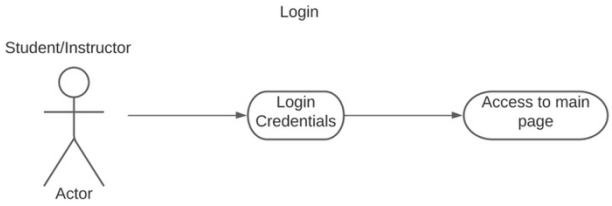


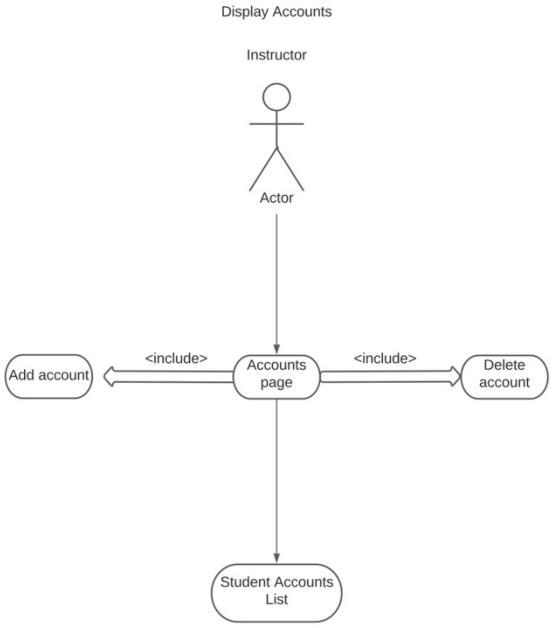
4.6 Use-Case 2's: Sequence Diagram

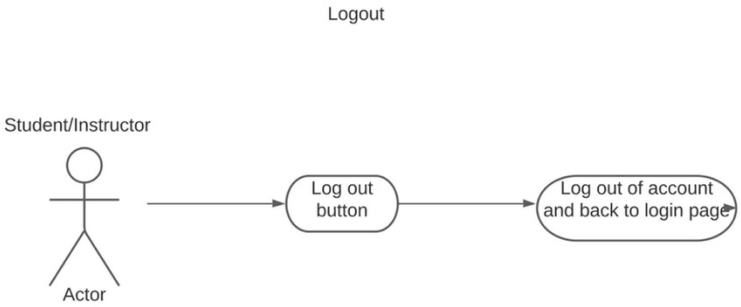
The use case will depend on a Timer class to track the study hours of the student, as the student starts with studying the material or doing some tasks the timer would start by the help of the startTimer() function, and after a long period when the student stops the timer would also stop by the help of the stopTimer() function. Thus, the hours spent will be calculated. After that a report will be generated for both the student and the instructor showing the hours that the student has studied at different time slots. Finally, The notification class would be used by the instructor to notify the students if they're late or on track with the material.



4.7 Use-Case Scenario 3







5. VLA Structure

5.1 UML Class

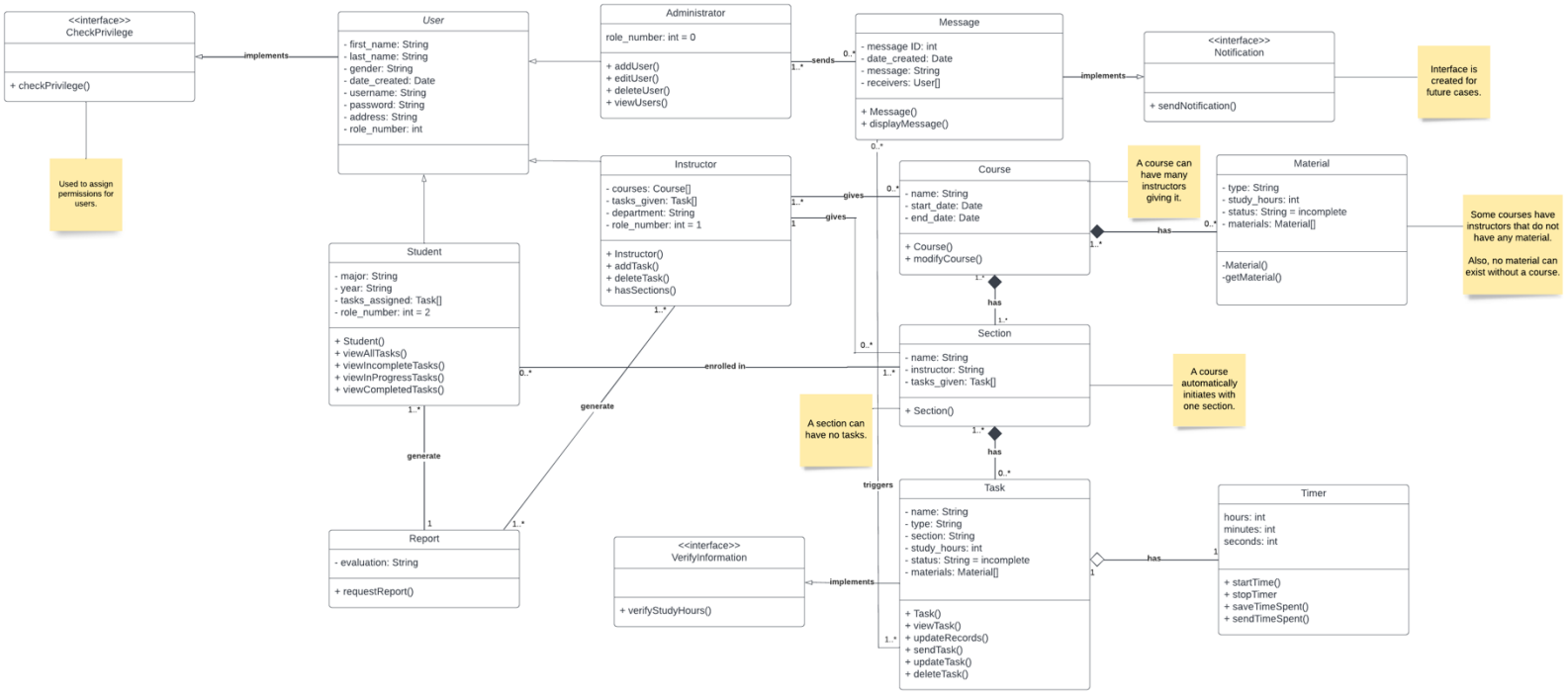


Figure 7: UML Class Diagram

The UML Class Diagram provides a more detailed look on the static view. The object-oriented organization of the system or the structure of the data processed by the system. Hence, the UML Class diagram shows the relationships between these classes. First we have the abstract class User which extends to create Instructor, Student, and Administrator. The User class also implements the interface CheckPrivileges which prevents the Student and Instructor from creating users (since they are all the same class) by looking up their role_number. Administrators are assigned 0 (highest), Instructors 1 (medium), Student 2 (lowest). Then Administrators can exhibit the functionalities previously discussed in the Use-Case Diagram (add, remove, modify user, etc.).

The Instructor can manage tasks hence they interact with the courses (which they give). The courses are composed sections. The Section contains Tasks and Material. The Tasks have a Timer. Hence, the Task triggers the Timer when the Material in the Section is accessed. Also the Task contains VerifyInformation interface that is implemented to verify information. In this case, we need to verify that

study hours complies with user requirements (Study hours have to be greater than 0). When the Task is triggered it interacts with the Message. The Message is generated and implements the Notification Interface (which enables the message to be sent by the system). The Message is also accessed by the Administrator to send notifications to the users in case of system disruptions, etc. Student class is enrolled in a course with a certain section. Finally, the Student and the Instructor can generate reports in order to view their progress (for Student), or for the Section (for the Instructor). The Instructor can also view the student's enrolled in their section individually.

5.2 State Diagram

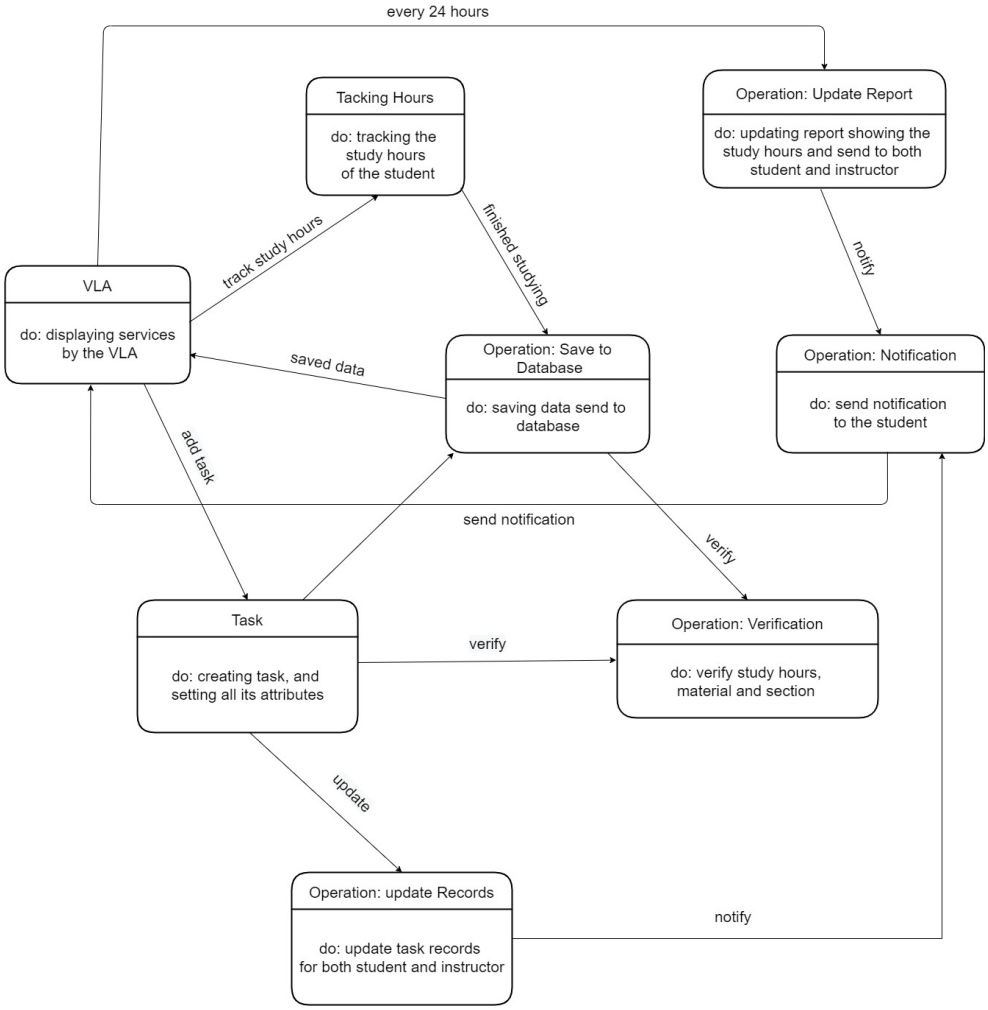


Figure 8: State Diagram

6. Conclusion

This Design and Architecture document shall serve to pave a clear pathway for Op-Engineers next phases. The document included different models to depict the system visually clearly for Engineers and Stakeholders alike. The Software Design and Architecture document entailed a visual blueprint of the system through Context Model, Architecture Diagram, Interaction Model, Behavioral Model, and Structural Model. Different models served different purposes. Context Diagram and Architecture Diagram provided high-level overview of the system. Meanwhile, the Interaction, Behavioral, and Structural provided different perspectives of the system from the inside. This design shall serve to be the cornerstone in setting an easy and smooth implementation and testing phase as it provides a strong pictorial description identifying the needed classes and tools to make the system functional/deployed. Also, it provided the system goals and desired behavior in a technical manner enabling testers to deploy tests that will reflect the previously natural language goal into a workable function in the VLA system.