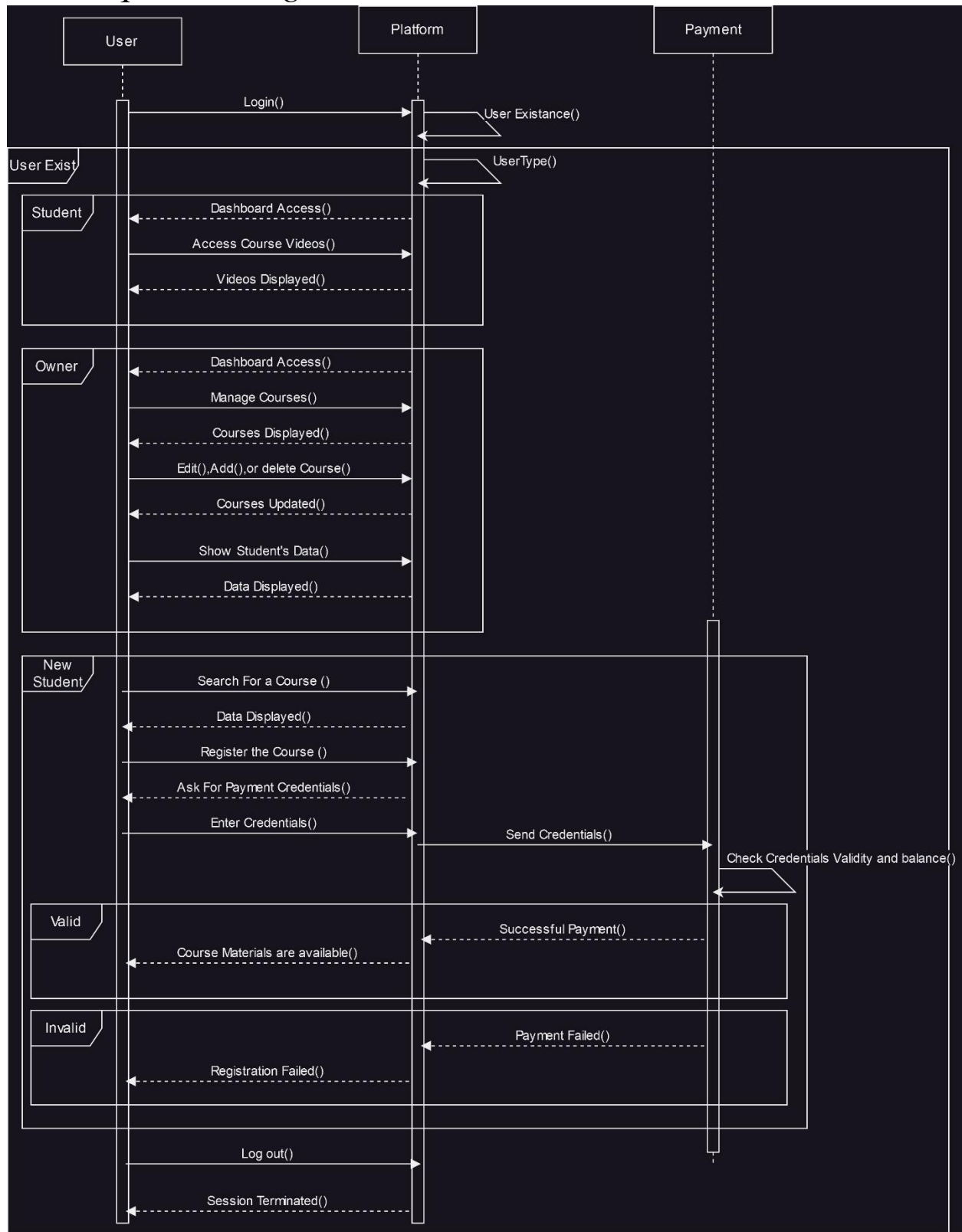# Brainstorm Task

1- *Software Requirements and Functionalities:*

- <u>Authentication:</u> The platform should have a user authentication system that would allow new students to register and the current students to login.
- <u>Course Management:</u> The platform should allow the owner to add, edit, and delete courses and to specify the course schedule, lessons, tutor.
- <u>Student Management:</u> The platform should allow the owner to manage students' data, including their progress in the courses, attendance, and course information.
- <u>Dashboard:</u> The platform should have a dashboard that can be accessed by the student and the owner.
- <u>Video Access:</u> The platform should allow current students to access the course videos.
- <u>Notification System:</u> The platform should have a Notification System to send remainders to students about lessons, assignments, and quizzes.
- <u>Payment System:</u> The platform should have a payment system to allow the students to pay for their courses.

2- Sequence Diagram:

User — **Platform** — **Payment**

Login()

User Existance()

User Exist

UserType()

Student

Dashboard Access()

Access Course Videos()

Videos Displayed()

Owner

Dashboard Access()

Manage Courses()

Courses Displayed()

Edit(),Add(),or delete Course()

Courses Updated()

Show Student's Data()

Data Displayed()

New Student

Search For a Course ()

Data Displayed()

Register the Course ()

Ask For Payment Credentials()

Enter Credentials()

Send Credentials()

Check Credentials Validity and balance()

Valid

Successful Payment()

Course Materials are available()

Invalid

Payment Failed()

Registration Failed()

Log out()

Session Terminated()

3- UML – Diagram:

Notification

- message: String
- recipient: User
- date:Date

+ Send(): void
+ View(): void

1..*   1..*

User

- name: String
- email: String
- password:String

+ login(): void
+ logout(): void

1..*
make

Payment

- amount: double
- status: bool

+ makePayment(credentials):
void

1

Extends   Extends

Owner

+ addCourse(courseId): void
+ editCourse(courseId): void
+ deleteCourse(courseId): void
+ studentData(studentId):Student

Student

- courseEnrolled: Course
- courseProgress: int
- attendance:int
- paymentStatus:bool

+ registerCourse(courseId): void
+ accessCourse Videos():
List<Videos>

Accessed by
1..*   1..*
Access

Video

- videoName: String
- videoId: String
- courseId: String
- tutor:String
- url:String

+ downloadVideo(videoId): void

Managed by   1

Registered by   1..*

1..*
Manages

Register
1

Course

- courseName: String
- courseId: String
- schedule: Date
- tutor:String
- studentsEnrolled:List<Student>
- coursePrice: double

+ AccessVideo(videoUrl): Video