**Problem Statement 9:**

Scenario:

The shift toward digital learning has opened up new opportunities for students but has also introduced significant challenges. Many students struggle with staying organized, motivated, and engaged in their studies when they lack the structure of a traditional classroom environment.

**Managing multiple assignments, keeping track of deadlines, and finding reliable study resources** can be overwhelming, particularly for students who are not accustomed to self-directed learning.

Additionally, the absence of in-person interactions with teachers and peers can lead to feelings of isolation, further diminishing **students' motivation and performance**.

While learning management systems (LMS) provide a basic framework for digital education, they often **lack the personalization and support needed to address these individual challenges**

**External API**

**Dashboards full(student performance..also what teacher should explain ..full analysis for both student himself and to teacher abt student)**

**Google and github standards for colours**

**What project model followed(ajile or ?) if agile what functionality u used**

# Draft-1: -( 24/08/24)

Objectives: -

1. Managing Multiple Assignments (another tab at other side )
2. Keeping Track of deadline ( side tab )
3. Finding Reliable Study Resources (help me button..youtube links but wont open in youtube app to avoid distractions) (each subject we need to find the resources , research papers etc)
4. Motivation (badges to post on linkedin) (superman flying like interview bit)
5. Performance
6. Student – teacher and peer interaction (google meet + summarization)(also a chat module to chat with others)
7. Support (Chatbot)
8. Extra:- self care tips to improve concentration …flash cards on a topic using ai

Target Audience: -

Students of (college / school )

Students doesn’t need to enrol to any course …but they will automatically gets enrolled into subjects for each year.

Authorization permission:-

1. For students to
   1. Access and Upload assignment documents
   2. Access the notes and resources
   3. Chat with peers and teachers and video call
2. For teachers to
   1. Access the students assignments and allot marks
   2. Upload the notes and resources
   3. Chat with students and video call with students

Scope of project

Technologies used

High level design document (Use case diagram)

And low level design document(component diagrams)

Functional specifications(Sequence diagrams..various interactions)

Wire frame diagram(instead of uml)

**Resources and tools:-**

1. <https://www.taskade.com/spaces/ZUiRPRtw9M3u2gJJ/?onboarding-open-generate-project=true>
2. [Elearning Management System (youtube.com)](https://www.youtube.com/watch?v=-s_-_Tz866w) (Created only admin..not for student and teacher)

# Day-2(25-08-24)

My idea:-

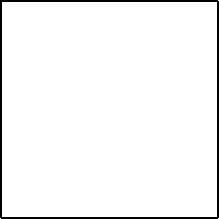
STUDENT:-  
  
Assignments  
 upload   
 submit   
 delete  
 check assignment  
 check score  
Chat   
 With friends  
 With Professor  
Personal Ai tutor  
Visual Elements of there course assignment scores  
deadlines  
all assignments  
Sticky note  
Notifications

Resources  
 View / check

download

TEACHER:-  
Assignments  
 upload question(in doc)   
 delete question   
 post assignment  
 check the student assignment  
 score the assignment  
Chat   
 with student  
Notifications

Resources  
 upload resources  
 delete resources  
 provide direct YouTube links



Visualization component(all assignments and their scores in one graph of a student)

Resources

Upload

Delete docuements

# DAY-3(26-08-24):-

Resources:-

1. <https://canvasjs.com/angular-charts/>
2. <https://canvasjs.com/docs/charts/integration/angular/>

Tasks Done:-

1. Created a bar graph ..(dashboard component)
2. Adjusted the height and width
3. Removed the water mark
4. Tried deploying phi but waste since it will be costly

Notes(On water mark removal and angular):-

* Used the setTimeOut …to delay the credit or watermark ..why ..without this the water mark wasn’t getting hidden
* Also when ever we reload we see that for a moment(blink of eye)
* We are using that because we want our chart to be loaded..
* A microservice is not just one api
* A microservice is a collection of api’s
* Dodnot touch index.html ..use onlyapp-component

**Code:-**

***Inside ts (chart + remove credit)(inside class)***

import { CanvasJSAngularChartsModule } from '@canvasjs/angular-charts';

  constructor() { }

  ngOnInit(): void {

    setTimeout(() => {

      const creditLink = document.querySelector('a[href="https://canvasjs.com/"]') as HTMLElement;

      if (creditLink) {

        creditLink.style.display = 'none';

      }

    }, 0.1); // Delay to ensure the chart is fully loaded

  }

  chartOptions = {

    title: {

      text: "Course-1 Report"

    },

    data: [{

      type: "column",

      dataPoints: [

        { label: "Assignment-1",  y: 10  },

        { label: "Assignment-2", y: 15  },

        { label: "Assignment-3", y: 25  },

        { label: "Assignment-4",  y: 30  },

        { label: "Assignment-5",  y: 28  }

      ]

    }]

  };

**Inside html of dashboard component:-**

 <div>

  <canvasjs-chart [options]="chartOptions"  ></canvasjs-chart>

</div>

# **Microservices:-**

1. Course
   * Display all courses(get …getAllCourses)
   * Recent courses accessed (get … getRecentCourses)
   * Visualise(get…getEachAssignmentScore)
2. Assignments
   1. Uploading assignments( post …uploadAssignment)
   2. Display assignment(get..displayAssignment)
   3. Delete assignment(delete…deleteAssignment)
   4. Deadlines ( get…getDeadline)
   5. Scoring..(get ..getScore)
   6. Submission Status(get..getSubmissionStatus)
   7. Files Submitted(get …filesSubmitted)
3. /\* Chat with peers and professors
   1. Send message(post ….sendMessage)
   2. Receive messages(get…receiveMessage) \*/
4. Personal Ai tutor
   1. Send the query or doubt (post…sendQuery)
   2. Get the doubt resolved (get …queryResult)
5. Profile Page
   1. Get the student details(get…getStudentDetails)
   2. Get the course details(get…getCourseDetails)(based on which student enrolled which courses)
   3. Visual elements (get….getTotalOfEachCourse)
6. Notification
   1. If deadline is over due or 1 day left u get notification (post…sendNotifications)
   2. Show the notifications(get …showNotifications)

/\* Resources(keep it static ..)

View resources(get…viewResources) \*/

**CLASS DIAGRAM:-**

**class diagram: -**

@startuml

class Course {

+getAllCourses()

+getRecentCourses()

+getEachAssignmentScore()

}

class Assignment {

+uploadAssignment()

+displayAssignment()

+deleteAssignment()

+getDeadline()

+getScore()

+getSubmissionStatus()

+filesSubmitted()

}

class PersonalAITutor {

+sendQuery()

+queryResult()

}

class ProfilePage {

+getStudentDetails()

+getCourseDetails()

+getTotalOfEachCourse()

}

class Notification {

+sendNotifications()

+showNotifications()

}

Student --> Course

Student --> Assignment

Student --> PersonalAITutor

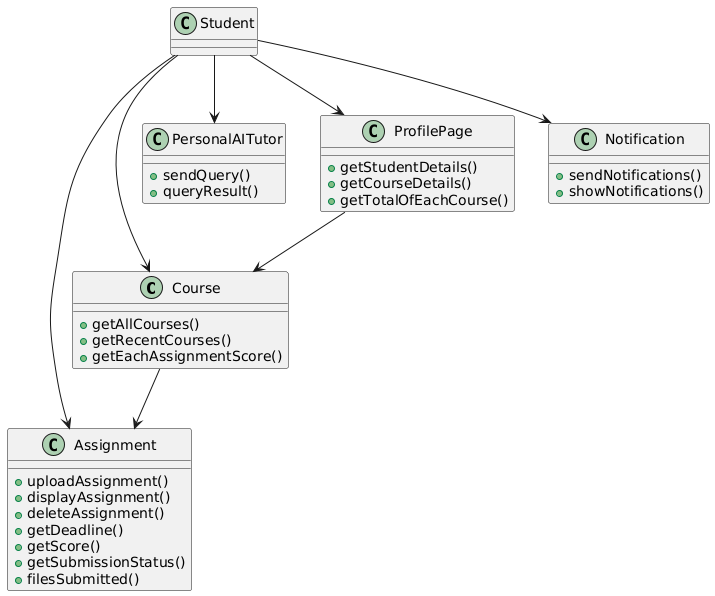
Student --> ProfilePage

Student --> Notification

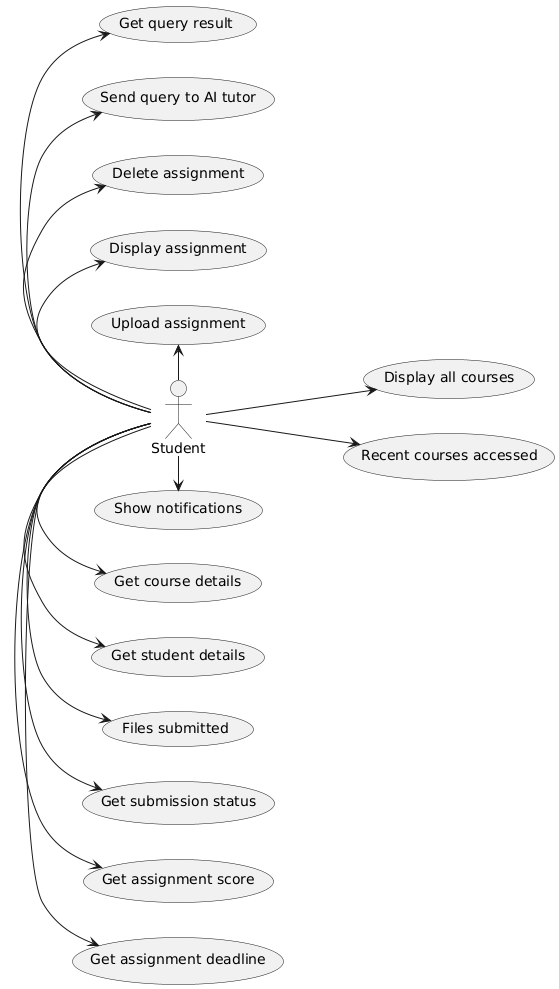
Course --> Assignment

ProfilePage --> Course

@enduml



**USE CASE DIAGRAM:-**



@startuml

left to right direction

actor Student

Student -down-> (Display all courses)

Student -down-> (Recent courses accessed)

Student -left-> (Upload assignment)

Student -left-> (Display assignment)

Student -left-> (Delete assignment)

Student -right-> (Get assignment deadline)

Student -right-> (Get assignment score)

Student -right-> (Get submission status)

Student -right-> (Files submitted)

Student -left-> (Send query to AI tutor)

Student -left-> (Get query result)

Student -right-> (Get student details)

Student -right-> (Get course details)

Student -right-> (Show notifications)

@enduml

# ER-Database Diagram:-

@startuml

entity "student" {

+id : int

--

name : varchar

password : varchar

email : varchar

courseId : int

}

entity "professor" {

+id : int

--

name : varchar

password : varchar

email : varchar

courseId : int

}

entity "admin" {

+id : int

--

username : varchar

name : varchar

password : varchar

email : varchar

}

entity "courses" {

+id : int

--

name : varchar

courseId: int

}

entity "resources" {

+id : int

--

title : varchar

type : varchar

filePath : varchar

courseId : int

}

entity "assignments" {

+id : int

--

title : varchar

description : text

dueDate : date

courseId : int

marks:int

}

' Relationships

student ||--o{ courses : "enrolls"

professor ||--o{ courses : "teaches"

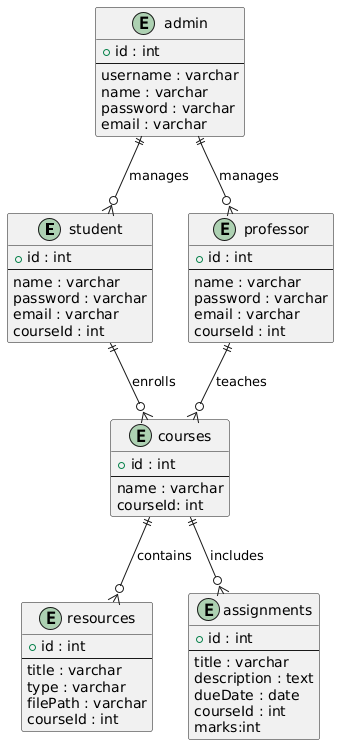
admin ||--o{ student: "manages"

admin ||--o{ professor: "manages"

courses ||--o{ assignments : "includes"

courses||--o{ resources: "contains"

@enduml



**Microservices Architecture:-**

**A diagram of a computer program

Description automatically generated**

# Class diagram -2(needs editing still):

@startuml

!define RECTANGLE class

RECTANGLE Student {

+ getAllCourses()

+ getRecentCourses()

+ getEachAssignmentScore()

+ getStudentDetails()

+ getCourseDetails()

+ getTotalOfEachCourse()

}

RECTANGLE Courses {

+ getAllCourses()

+ getRecentCourses()

+ getEachAssignmentScore()

}

RECTANGLE Assignments {

+ uploadAssignment()

+ displayAssignment()

+ deleteAssignment()

+ getDeadline()

+ getScore()

+ getSubmissionStatus()

+ filesSubmitted()

}

RECTANGLE PersonalAiTutor {

+ sendQuery()

+ queryResult()

}

RECTANGLE Notification {

+ sendNotifications()

+ showNotifications()

}

Student --> Courses : interacts

Student --> Assignments : interacts

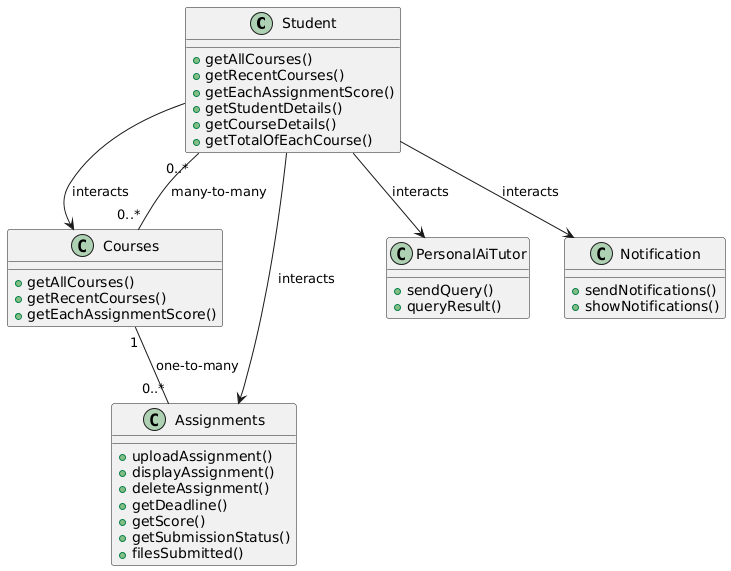
Student --> PersonalAiTutor : interacts

Student --> Notification : interacts

Courses "0..\*" -- "0..\*" Student : many-to-many

Courses "1" -- "0..\*" Assignments : one-to-many

@enduml



**Use case Diagram:**

@startuml  
left to right direction

actor Admin  
actor Teacher  
actor Student

rectangle "Digital Learning Platform" {

    (Manage Student Accounts) as MSA  
    (Manage Teacher Accounts) as MTA

    (Create & Manage Courses) as CMC  
    (Create & Assign Assignments) as CAA  
    (Upload & Organize Resources) as UOR

    (Access Courses & Materials) as ACM  
    (Manage Assignments) as MA  
    (Track Personal Progress) as TPP  
    (Receive Notifications & Reminders) as RNR  
}

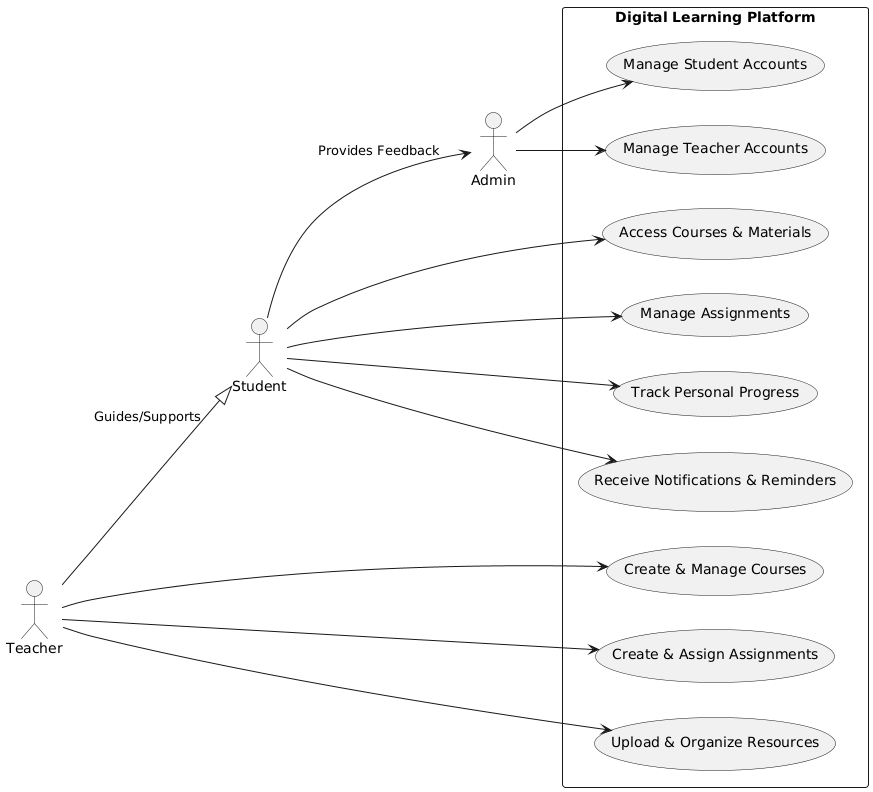
Admin --> MSA  
Admin --> MTA

Teacher --> CMC  
Teacher --> CAA  
Teacher --> UOR

Student --> ACM  
Student --> MA  
Student --> TPP  
Student --> RNR

' Define interactions/relationships between roles  
Teacher --|> Student : Guides/Supports  
Student --> Admin : Provides Feedback

@enduml

\

**Sequence diagram:-**

@startuml  
actor Admin  
actor Teacher  
actor Student

participant "Account Management Service" as AMS  
participant "System Configuration Service" as SCS  
participant "Course Management Service" as CMS  
participant "Assignment Management Service" as AMS2  
participant "Learning Management Service" as LMS  
participant "Notification Service" as NS

== Admin Interactions ==  
Admin -> AMS : Manage Student Accounts  
AMS -> Admin : Confirmation

Admin -> AMS : Manage Teacher Accounts  
AMS -> Admin : Confirmation

Admin -> SCS : Configure System  
SCS -> Admin : Confirmation

== Teacher Interactions ==  
Teacher -> CMS : Create & Manage Courses  
CMS -> Teacher : Confirmation

Teacher -> AMS2 : Create & Assign Assignments  
AMS2 -> Teacher : Confirmation

Teacher -> AMS2 : Upload & Organize Resources  
AMS2 -> Teacher : Confirmation

== Student Interactions ==  
Student -> LMS : Access Courses & Materials  
LMS -> Student : Provide Materials

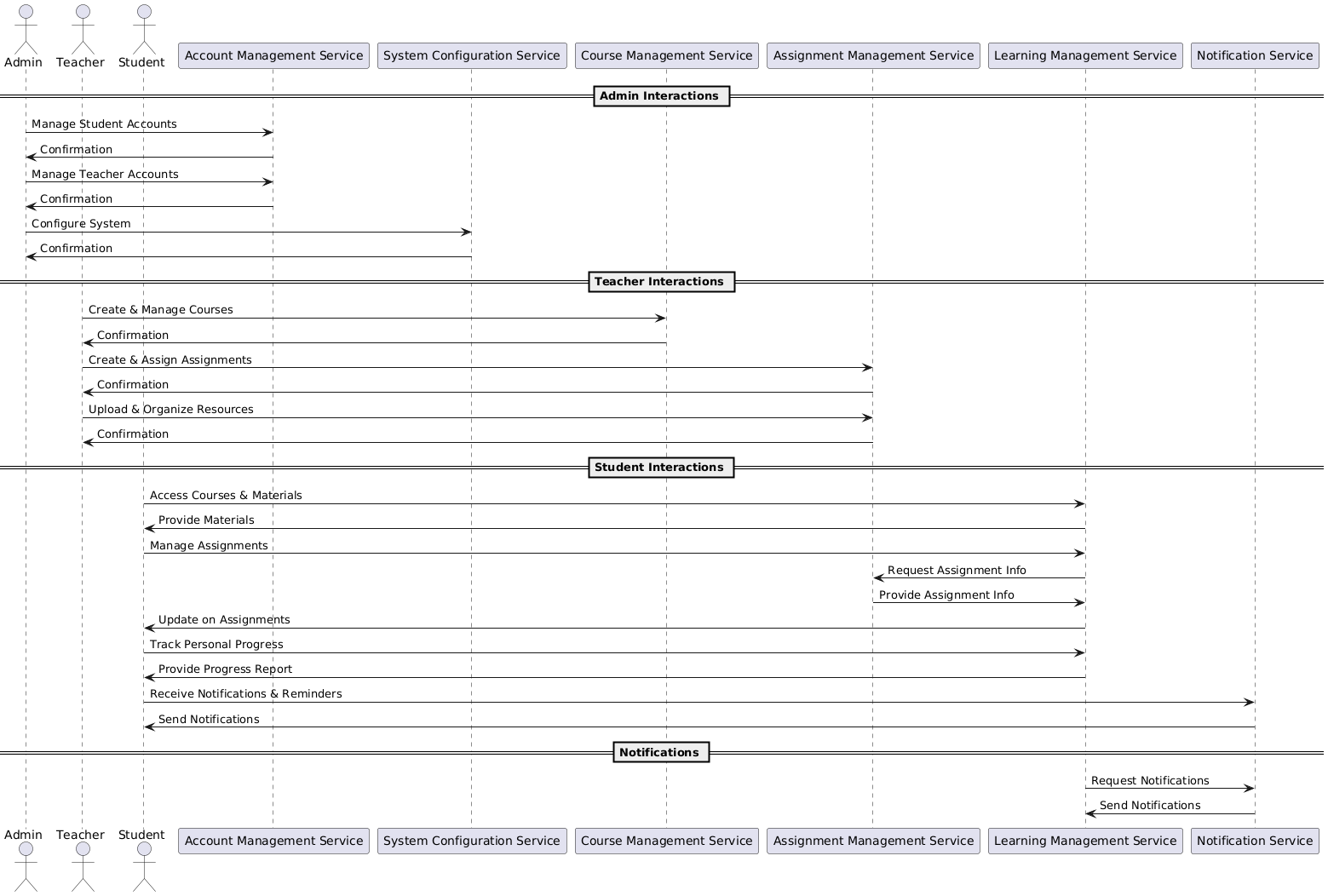
Student -> LMS : Manage Assignments  
LMS -> AMS2 : Request Assignment Info  
AMS2 -> LMS : Provide Assignment Info  
LMS -> Student : Update on Assignments

Student -> LMS : Track Personal Progress  
LMS -> Student : Provide Progress Report

Student -> NS : Receive Notifications & Reminders  
NS -> Student : Send Notifications

== Notifications ==  
LMS -> NS : Request Notifications  
NS -> LMS : Send Notifications

@enduml



**CLASS DIAGRAM(Final):-**

@startuml  
left to right direction

package "User Management Service" {  
    class User {  
        +String userId  
        +String username  
        +String email  
        +String passwordHash  
        +String role  
        +updateUser(userId: String, username: String, email: String, password: String) : void  
        +deleteUser(userId: String) : void  
        +getUser(userId: String) : User  
    }

    class Admin {  
        +String adminId  
        +String adminName  
        +createUser(user: User) : void  
        +deleteUser(userId: String) : void  
    }

    class Teacher {  
        +String teacherId  
        +String teacherName  
        +assignCourse(courseId: String) : void  
    }

    class Student {  
        +String studentId  
        +String studentName  
        +Date dateOfBirth  
        +enrollCourse(courseId: String) : void  
        +submitAssignment(assignmentId: String, content: String) : void  
    }

    User <|-- Admin  
    User <|-- Teacher  
    User <|-- Student  
}

package "Course Management Service" {  
    class Course {  
        +String courseId  
        +String title  
        +String description  
        +String teacherId  
        +createCourse(title: String, description: String, teacherId: String) : void  
        +updateCourse(courseId: String, title: String, description: String) : void  
        +getCourse(courseId: String) : Course  
    }  
}

package "AiTutor Service" {  
    class AiTutor {  
        +String receiveQueries(query: String) : String  
        +String provideOutput(query: String) : String  
    }  
    Student "1" -- "0..\*" AiTutor : communicates  
}

package "Assignment Management Service" {  
    class Assignment {  
        +String assignmentId  
        +String title  
        +String description  
        +Date dueDate  
        +String courseId  
        +String teacherId  
        +createAssignment(title: String, description: String, dueDate: Date, courseId: String, teacherId: String) : void  
        +updateAssignment(assignmentId: String, title: String, description: String) : void  
        +deleteAssignment(assignmentId: String) : void  
        +getAssignment(assignmentId: String) : Assignment  
    }

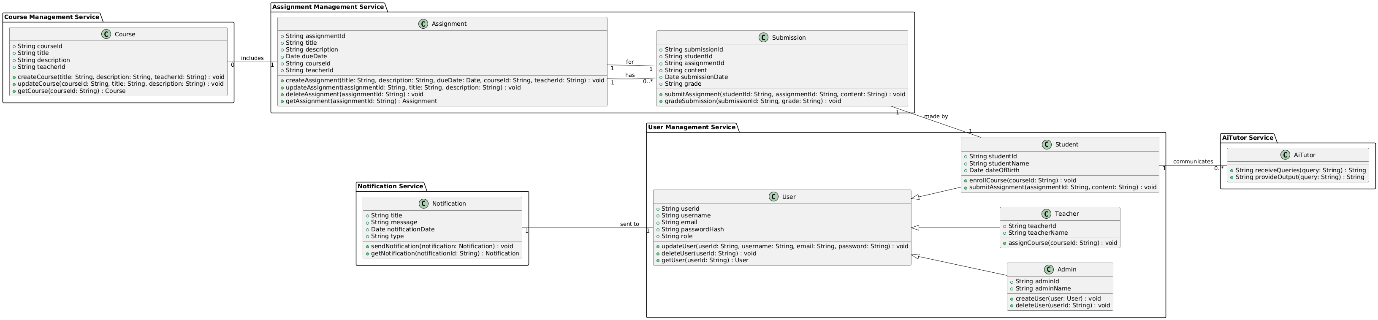
    class Submission {  
        +String submissionId  
        +String studentId  
        +String assignmentId  
        +String content  
        +Date submissionDate  
        +String grade  
        +submitAssignment(studentId: String, assignmentId: String, content: String) : void  
        +gradeSubmission(submissionId: String, grade: String) : void  
    }

    Assignment "1" -- "0..\*" Submission : has  
    Submission "1" -- "1" Student : made by  
    Submission "1" -- "1" Assignment : for  
    Course "0" -- "1" Assignment : includes  
}

package "Notification Service" {  
    class Notification {  
        +String title  
        +String message  
        +Date notificationDate  
        +String type  
        +sendNotification(notification: Notification) : void  
        +getNotification(notificationId: String) : Notification  
    }

    Notification "1" -- "1" User : sent to  
}

@enduml



**Component Diagram: -**

@startuml  
!define RECTANGLE\_COMPONENT  
!define RECTANGLE\_COMPONENT(a\_alias, a\_label, a\_text\_color, a\_back\_color, a\_border\_color)

left to right direction

package "User Management Service" {  
    [User Service] as UserService  
    [Admin Service] as AdminService  
    [Teacher Service] as TeacherService  
    [Student Service] as StudentService  
}

package "Course Management Service" {  
    [Course Service] as CourseService  
}

package "AiTutor Service" {  
    [AiTutor Service] as AiTutorService  
}

package "Assignment Management Service" {  
    [Assignment Service] as AssignmentService  
    [Submission Service] as SubmissionService  
}

package "Notification Service" {  
    [Notification Service] as NotificationService  
}

UserService -[hidden]down- AdminService  
UserService -[hidden]down- TeacherService  
UserService -[hidden]down- StudentService

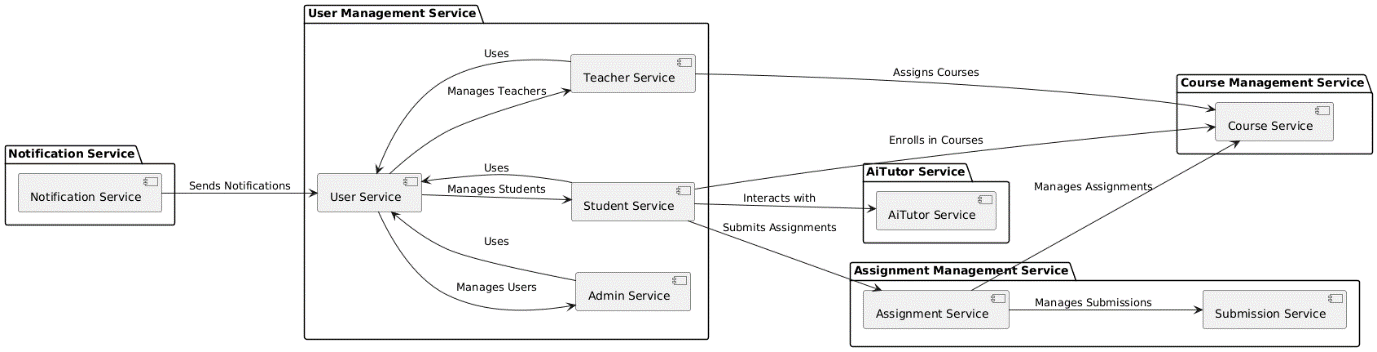
UserService --> AdminService : Manages Users  
UserService --> TeacherService : Manages Teachers  
UserService --> StudentService : Manages Students

AdminService --> UserService : Uses  
TeacherService --> UserService : Uses  
StudentService --> UserService : Uses

StudentService --> CourseService : Enrolls in Courses  
TeacherService --> CourseService : Assigns Courses

StudentService --> AssignmentService : Submits Assignments  
AssignmentService --> SubmissionService : Manages Submissions  
AssignmentService --> CourseService : Manages Assignments

StudentService --> AiTutorService : Interacts with  
NotificationService --> UserService : Sends Notifications  
@enduml



@startuml

!define RECTANGLE\_COMPONENT

!define RECTANGLE\_COMPONENT(a\_alias, a\_label, a\_text\_color, a\_back\_color, a\_border\_color)

left to right direction

package "User Management Service" {

[User Service] as UserService

[Admin Service] as AdminService

[Teacher Service] as TeacherService

[Student Service] as StudentService

}

package "Course Management Service" {

[Course Service] as CourseService

}

package "AiTutor Service" {

[AiTutor Service] as AiTutorService

}

package "Assignment Management Service" {

[Assignment Service] as AssignmentService

[Submission Service] as SubmissionService

}

package "Notification Service" {

[Notification Service] as NotificationService

}

UserService -[hidden]down- AdminService

UserService -[hidden]down- TeacherService

UserService -[hidden]down- StudentService

UserService --> AdminService : Manages Users

UserService --> TeacherService : Manages Teachers

UserService --> StudentService : Manages Students

AdminService --> UserService : Uses

TeacherService --> UserService : Uses

StudentService --> UserService : Uses

StudentService --> CourseService : Enrolls in Courses

TeacherService --> CourseService : Assigns Courses

StudentService --> AssignmentService : Submits Assignments

AssignmentService --> SubmissionService : Manages Submissions

AssignmentService --> CourseService : Manages Assignments

StudentService --> AiTutorService : Interacts with

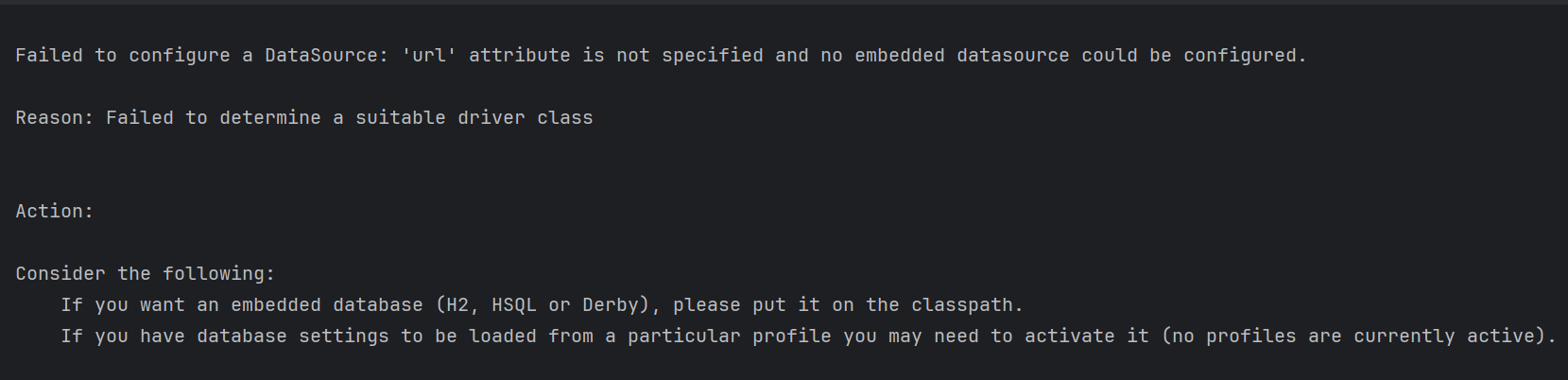
NotificationService --> UserService : Sends Notifications

@endum

@enduml

# DAY-4(27/08/24): -

**Problem #1:**



**Solution:-**

It is because we are adding the jdbc or h2 or jpa or mysql etc etc in the pom.xml

Just add this line in application.properties

**spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration**

**Refer:-** [Resolving Failed to Configure a DataSource Error | Baeldung](https://www.baeldung.com/spring-boot-failed-to-configure-data-source)

When using spring security in pom.xml it shows a login page directly

**Default username** is “**user**” **password** is **generated** in the cmd …u can configure u r own username and password ..will show **bad** **credentials** on screen if wrong username and password else ….**whitelabel** **error** (if not given any fallback method or endpoint)..if given endpoint or done some coding …then should show according to that

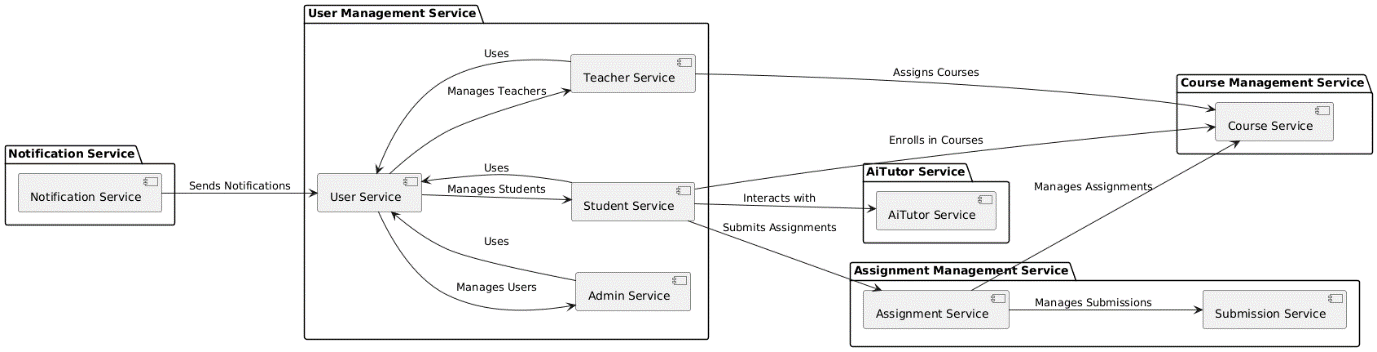
“”” spring.security.user.name=sergey

spring.security.user.password=sergey

spring.security.user.roles=manager “”” in application.properties

Refer:- [Spring Security Default Username, Password, Role - Apps Developer Blog](https://www.appsdeveloperblog.com/spring-security-default-username-password-role/#:~:text=spring.security.user.name%3Dsergey,spring.security.user.password%3Dsergey%20spring.security.user.roles%3Dmanager)

**Component Diagram: -**



**ENTITY RELATION SHIP Diagram:-**

**Code:-**

**Website:- https://dbdiagram.io/d**

// Use DBML to define your database structure

// Docs: https://dbml.dbdiagram.io/docs

Table StudentTable {

  studentId integer [primary key]

  studentName varchar

  courseIds varchar  // storing list of studentIds as a comma-separated string

  age integer

  email varchar

  phoneno varchar

  courseId integer

  username varchar

  password varchar

  gender varchar

}

Table ProfessorTable {

  professorId integer [primary key]

  professorName varchar

  department varchar

  email varchar

  experience integer

  gender varchar

  phoneno varchar

  password varchar

}

Table CourseTable {

  courseId integer [primary key]

  courseName varchar

  studentIds varchar // storing list of studentIds as a comma-separated string

  assignmentIds varchar

  description text

  professorId integer

  semester varchar

}

Table AssignmentTable {

  assignmentId integer [primary key]

  professorId integer

  courseId integer

  title varchar

  marks integer

  fileName varchar

  deadline timestamp

}

Ref: CourseTable.professorId > ProfessorTable.professorId // many-to-one

Ref: AssignmentTable.courseId > CourseTable.courseId // many-to-one

Ref:StudentTable.courseId>CourseTable.courseId

Ref: CourseTable.assignmentIds< AssignmentTable.assignmentId

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedIn black and white

A screenshot of a computer

Description automatically generatedshowing many to one from student to course

A screenshot of a computer

Description automatically generated

# 03-09-24:-

A screenshot of a computer

Description automatically generated

@startuml

actor Admin

actor Professor

actor Student

participant "Course Management Service" as CMS

participant "Assignment Management Service" as AMS

participant "User Management Service" as UMS

== Admin Interactions ==

Admin -> UMS : Manage Students and professor Accounts

== Teacher Interactions ==

Professor -> CMS : Create & Manage Courses

Professor-> AMS : Uploads assignments

Professor -> CMS : Upload Resources

== Student Interactions ==

Student -> CMS : Access Courses & Materials

CMS -> Student : Provides Resources & Materials

Student-> AMS: Can View Assignment

AMS->Student: Provides Assignments

@enduml