Title of the Project:

Online Learning Management System

Brief description of the project:

Organization Details:

The organization we are targeting for the project is a mid-size educational institution, which is expanding its scope to include online learning. The Institute is going to target large international userbase from various geographical regions. The institute's current structure is of traditional nature with focus on in person classes, and manual student management process. The system will be designed for the institution to improve its reach and student management process.

Project Overview:

The main aim of this system will be to make the learners and teachers life easier to organise the contents, assignments, grading and gaining feedback on each assignment. This gives a personal touch with a collaborative learning. Additionally, this system will be easily accessible to everyone from any location.

Goals:

1. **Course Creation and management:** Instructors will be able to create courses, upload study materials, create quizzes, tests and do online assessments.

2. **Student Enrollment:** System will be able to automate the student enrollment and onboarding. Students will be included in person as well as online students.

3. **Performance tracking:** Students as well as Instructors will be able to monitor their progress, grades, feedback.

4. **Assessments and exams:** Instructors could assign students assignments, quizzes and perform online exams in a secure manner.

5. **Integration with Technology:** The system will have integrated virtual technologies such as Google Meets, Teams, and incorporate use of AI. The system can also be integrated with other online learning platforms to create collaborative courses.

6. **Future scope:** System could be modified in future to provide the corporate employees with employer specific training.

The rationale for the project:

Business need:

With the digitally grown world, it is important to make the educational system effective. Sometimes, it becomes difficult for the instructors to manage each student's work and assessments. Moreover, there are challenges faced by students and instructors in on-campus learning. Some of those challenges are:

- Manual grading for quizzes and assignments
- Limited materials provided in class
- Communication gap
- Limited feedback for the student work
- Scalability and engagement progress

Due to these factors, it is essential to design a system that will be capable of handling real-time scenarios for example, instant grading, automatic assignments for student work, submission of the assignment, providing feedback, open communication channels, discussion forums, and more.

Improving or automating business processes:

Business Process Automation (BPA):

- **Automation:** Student enrolment, course assignment, progress monitoring, grading, and certification issuance are just a few of the manual operations that the LMS automates. When completed manually, these jobs usually involve a lot of human effort, which is also prone to mistakes and inefficiencies.

- **Examples:**
- Automated monitoring of student performance and progress.
- Certificates and compliance reports are automatically generated after necessary training is finished.
- Alerts and reminders regarding upcoming assessments, and assignments.

Business value:

Cost Efficiency:

- **Lowering Infrastructure costs:** Institutions can reduce infrastructure costs such as space requirements, canteen, hostels, manpower, and maintenance costs.
- **Scalability:** The LMS scales easily without requiring major new infrastructure or additional administrative resources, regardless of whether the Institution grows its student base to an international level.

Enhanced Compliance and Risk Mitigation:

- **Automated compliance tracking:** The LMS will make sure that learners finish the required material for the section of the course.

Faster Time-to-Market for New Learning Initiatives:

- **Rapid deployment:** The LMS enables the speedy delivery of new educational courses and programs, enabling learners to react more swiftly to evolving demands like market trends, new legislation, or technological advancements.

Better Learning Outcomes:

- **Personalized learning paths:** Learners receive instruction that is pertinent and related to their work functions or academic objectives.
- **Ongoing professional development:** The system will focus on both the beginners and the advanced learners. Instructors can supply materials along with a chance which will increase the growth and education along with development of the leaners using LMS. This will result in learners who are more capable and knowledgeable.

Data-Driven Decision Making:

- **Learning analytics:** Comprehensive reports and dashboards that provide information on learner progress, completion rates, and knowledge gaps will be made available by the LMS. Educators can use this information to inform data-driven decisions about content improvement or areas where students are having difficulty.
- **Performance tracking:** By connecting learning objectives to academic performance, organizations may evaluate how training affects learners' productivity and skill development.

Improved Learner Engagement and Retention:

- **Interactive learning experiences:** The LMS incorporates multimedia (simulations, movies, and quizzes) to make learning more dynamic and interesting, which enhances results and retention.
- **Gamification and individualized learning paths:** Leaderboards, badges, and tailored learning experiences inspire learners to learn and grow.

Analysis of requirements gathering for Online LMS:

Steps in requirements gathering:

1. **Stakeholder Identification:** Students, instructors, administrators.

2. **Identification Approach:** Use interviews, group sessions, and surveys to understand the pain points and expectations of each of these stakeholder groups.

3. **Problem Statement Definition:**
   - Key Challenges identified:
     - Manual processes like grading and enrolment.
     - Limitations in feedback mechanisms for student work.
     - Communication gap between students and instructors.
     - Limited scalability and accessibility for increasing user bases.

4. **Requirements gathering Techniques:**
   - **Interviews:** Explore needs and expectations by instructors and administrators to establish the functionalities for instance, automated grading, dashboards and feedback systems.
   - **Questionnaires:** Get the input of students over their preferred features about discussion forums, multimedia contents, and gamification.
   - **Observation:** Observe the current classroom environment and online learning to outline inefficiencies and areas that can be improved.
   - **Workshops:** Collaboration of stakeholders through workshops for prioritizing the requirements.

5. **Functional requirements:**
   1. **User Authentication:**
      1.1 System should allow learners, instructors to login.
      1.2 System should allow users to reset or recover the password.
      1.3 Admins should be allowed to login.
   2. **Manage Course:**
      2.1 Instructors should be able to schedule the courses.
      2.2 Instructors can cancel the course.
      2.3 Instructors can update the course.
      2.4 Instructors should be able to track the progress of the student.
      2.5 Instructors can create and assign the quizzes, modules and provide materials to learners.
   3. **Course enrolment of Learners:**
      3.1 System should be able to manually help to enrol learners or automatically, based on criteria.
      3.2 Learners can view their enrolled course.
   4. **Material delivery:**
      4.1 Instructor can provide materials such as PPT, books, video documents and links.
      4.2 Learners can submit the assignments, projects, quizzes and exams.
      4.3 System should allow learners to download the materials.
   5. **Assessment and Grading:**
      5.1 Provide automated grading for quizzes and produce immediate results.
      5.2 System should allow learners to submit the assigned tasks.
      5.3 Allow instructors to grade assignments manually and provide feedback.

   6. **Performance Tracking:**
      6.1 Student performance for course completions, grades, and participation.
      6.2 Provide analytics dashboards to instructors regarding class performance.
      6.3 Learners should be able to get the achievement reports.
   7. **Feedback and Communication:**
      7.1 System should be capable of including discussion forums for students and instructors.
      7.2 Learners should have an option to provide feedback about courses and instructors.

8. **Gamification and Engagement:**
   8.1 Award Badges and Certificates upon completion.
   8.2 Maintain leaderboards based on learning performance.
   8.3 Learners should get their achievement report which contains their leaderboard position, certificates and award badges.
9. **Administrative Features:**
   9.1 Send notifications to learners for upcoming assignments, due dates, and changes to the course via alerts.
   9.2 To provide only the required access to the users.
   9.3 System should be able to send the alert notifications if the system is experiencing a downtime.
   9.4 System should be able to handle payments.

6. **Non-Functional Requirements:**
   1. **Performance Requirements:**
      1.1 System should be able to handle at least 15,000 users.
      1.2 System should be able to load courses pages in less than 2 seconds.
   2. **Security Requirements:**
      2.1 Use encryption on sensitive data such as user credentials or grades.
      2.2 Two factor authentication for login.
   3. **Operations Requirements:**
      3.1 The system will operate with the latest versions of web browsers like Chrome, edge, safari.
      3.2 The system should be accessed on mobile devices and tablets.
      3.3 The system should be able to connect to printers to wirelessly.
   4. **Cultural and Political Requirements:**
      4.1 Adherence to copyright for usage and sharing of content.
      4.2 Compliance with institutional and governmental regulations for online education.
   5. **Scalability Requirements:**
      5.1 System should be scalable enough to handle an increasing number of users and courses as the institution expands.
   6. **Availability Requirements:**
      6.1 System should be up for most of the time without intervention.
      6.2 System should have an efficient backup mechanism to avoid loss of data.

Use Case Diagram:

Major Actors: Learner, Instructor, Administrator.

- **Learner:** Learner is capable to login, add course, submit assigned tasks, download materials, get the achievement reports and provide feedback.
- **Instructor:** An instructor is responsible to login, schedule course, cancel course, track progress of learners, assign tasks and to grade.
- **Administrator:** Admins can login, provide dashboard, notification alerts and restricts the access to unknown users.

Major Use-Cases:

Instructor: Instructor Login, Reset Password, Schedule Courses, Cancel Courses, Update Course, Assign Tasks, Provide Materials, Track Progress, Grading Assignment, Award Badges, Provide Feedback.

Learner: Learner Login, Add Courses, Recover Password, Payment Status, View Courses, Download Materials, Submit Assigned Tasks, Acheivement Report, Cousrse Feedback, LeaderBoard Position, Instructor Feedback.

Administration: Admin Login, Payment Status, Create Reports, Provide Alerts, Access Restriction.

Description of each category Use-Case:

Instructor:

- **Instructor Login:** Included use case where an Instructor will be able to Login.
- **Reset Password:** Instructor can change their password if they forget.
- **Schedule Courses:** Instructor will be able to schedule the courses.
- **Cancel Courses:** Instructor can remove or cancel the scheduled courses.
- **Update Courses:** Instructor can update the existing information for a course.
- **Assign Tasks:** Instructor can assign tasks to learners.
- **Provide Materials:** Instructor can provide learning materials.
- **Track Progress:** Instructor can track the progress of the learner.
- **Grading Assignment:** Instructor can grade the assignment to know where the learner stands.
- **Award Badges:** This will be an included case where Instructor can award badges to the students according to their performance.
- **Provide Feedback:** This will also be an included use case where the instructor can provide the feedback.

Learner:

- **Learner Login:** Included use case where a learner will be able to Login.
- **Add Courses:** Learner will be able to add the courses and will be able to see on the dashboard.
- **Recover Password:** Learners can recover their password if they forget.
- **Payment Status:** This will be a mandatory step as learners need to make payment if they wants to enroll to the course.
- **View Courses:** Learners can view the enrolled courses.
- **Download Materials:** This will be an included case where learners will be able to download the materials provided by the instructor for instance, a ppt, book, reference links etc.
- **Submit Assigned Tasks:** Learners needs to complete the assigned tasks by the instructor.
- **Achievement Report:** Learners will be able to get the report of their acheivement by the instructor.
- **Course Feedback:** Learners will be able to provide course feedback.
- **Instructor Feedback:** This will be an included case where the learners will be able to provide the feedback for the instructor who will be taking the course.
- **Leaderboard Position:** This will be an included case where learners will be able to see the leadearboard position as to understand where the learner stands in their acheivement report.

Administration:

- **Payment Status:** Admins will have the control to see whether the payments where done.
- **Create Reports:** Admins will be able to create the reports of number of enrollments to the courses.
- **Provide Alerts:** Admins will be able to send the alert notifications when the system will face the downtime. Admins will also provide remainder as in if there is any upcoming assignments, due dates and changes to course.
- **Access Restriction:** Admin will be able to control the access restrictions. Admins will give correct access to dedication persons.

Flow of events for 2 use cases:

Use Case: Submit Assigned Tasks.

Actors: Learner.

Purpose: Learner should be able to submit the assigned tasks for instance an assignment.

Overview: The learner gets the assigned task by the instructor. The learner completes the assignment and then submits the assignment. The notification will be given to the learner upon completion of the assignment.

Type: Essential.

Preconditions: The learner needs to log into the system. Assignment must be assigned and have the submission available.

Postconditions: The assignment will be saved after successful submission and the file remains stored in the system. Notification will be sent to the learner.

Special requirements: Learner should be notified in 10 seconds upon submission.

Normal flow:

1. Following a successful login, the learner accesses the "My Assignments" part of the system.
2. The system displays a list of assigned tasks for the learner.
3. The learner selects a task from the list.
4. The system displays the assignment information, including instructions and due date.
5. The learner clicks "Submit" after uploading their completed assignment file.
6. On successful submission, the system saves the submission and updates the learner's assignment status to "Submitted".
7. The system confirms to the learner through a notification.

Alternate flow of events:

1. If submission occurs after the due date then the system will give an error saying that the submission deadline has already passed. Contact the instructor and return to step 4.
2. If the learner submits the file in incorrect format then system throws an error of incorrect file format. Return to step 5.

Use case: Schedule courses.

Actors: Instructor.

Purpose: Instructor should be able to schedule the courses.

Overview: Instructor should be able to add the courses so that learners can view the start date and end date and then enrol themselves into the courses.

Type: Essential.

Preconditions: Instructor should be logged in. The system should have an access to course repository.

Postconditions: The course has been successfully arranged and is open to learners.

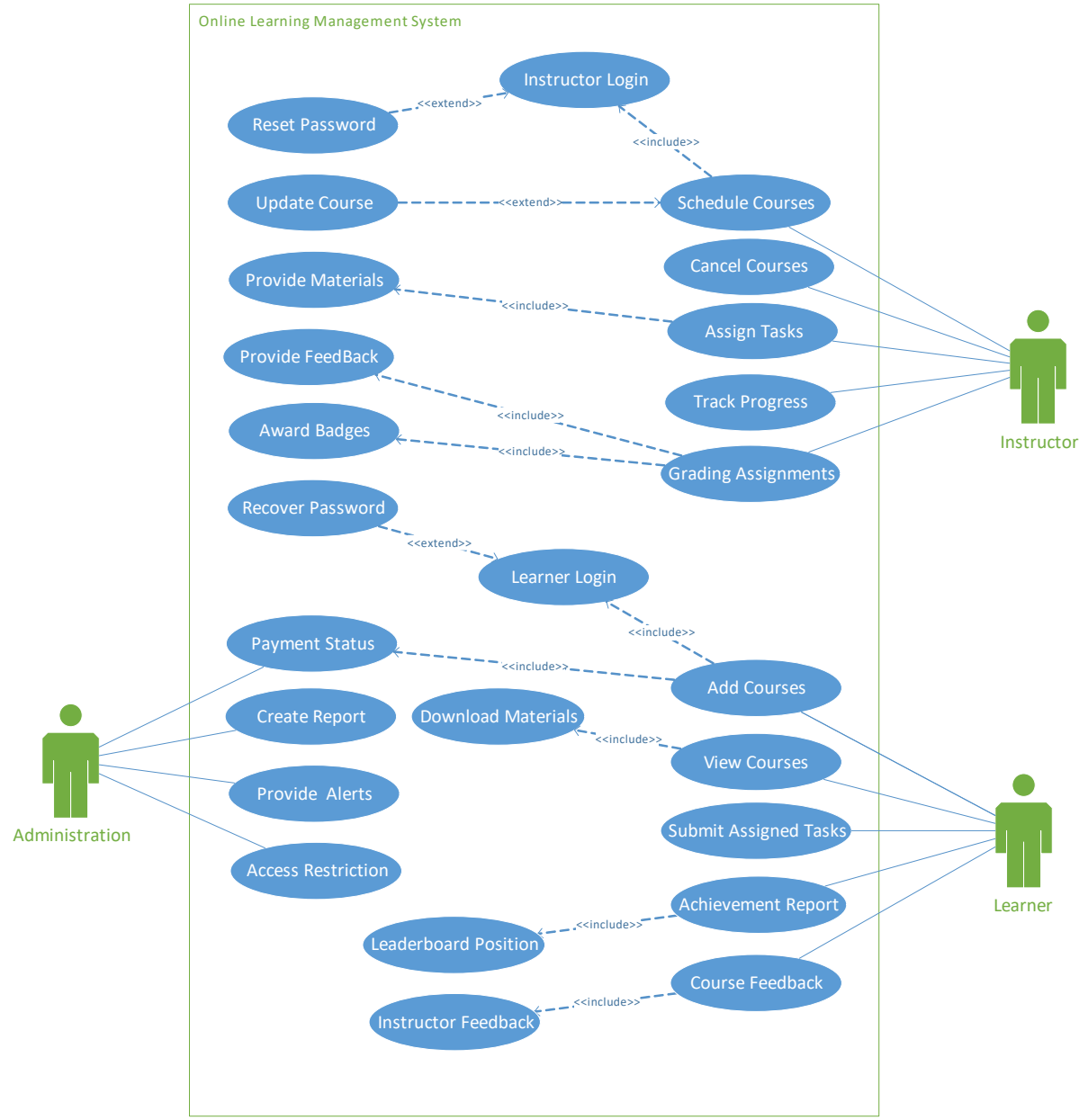
Normal Flow:

1. The instructor accesses the "Manage Course" area.
2. The system displays the list of existing courses and provides an option to create a new one.
3. The "Schedule Course" option is chosen by the instructor.
4. The instructor is prompted by the system to enter course information, including: name of the course, description, start and end dates, and target learners.
5. The request for the course schedule is made by the instructor.
6. Notify the respective students for the new course available for them.

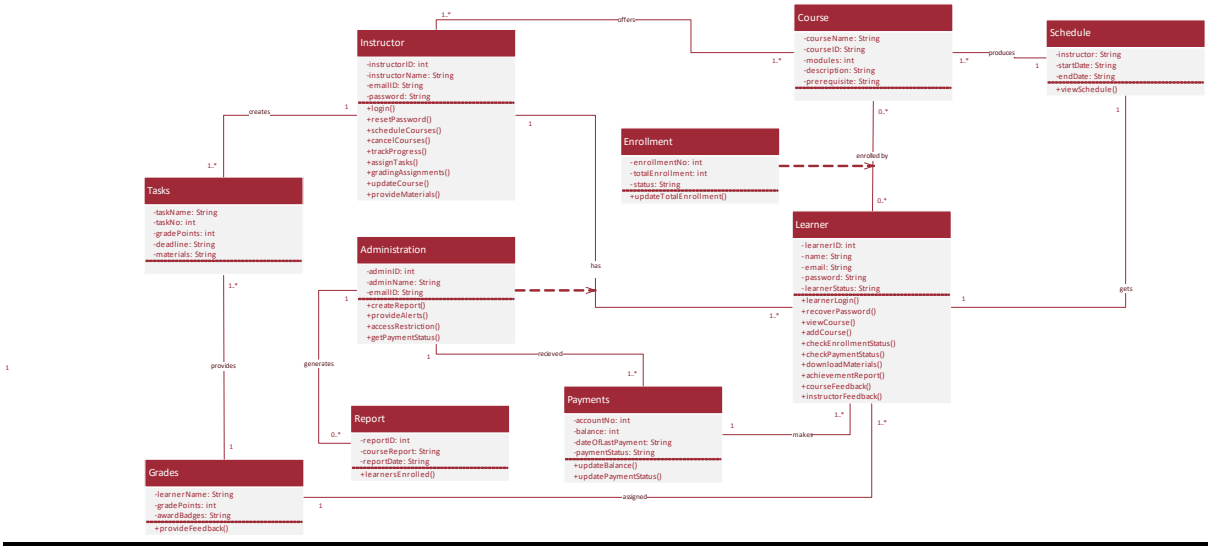
Alternate flow of events:

1. The system will prompt the instructor to review the data and shows an error message in case any of the information entered is wrong for instance, if the start date is in the past or course name is wrong. Return to step 4. Instructor edits data and resubmits it.

Use Case Diagram:



Class Diagram:



Domain classes:

1. **Instructor:** Instructor is capable to login, schedule, cancel, update courses, assign tasks, track progress of the learner and grade the assignments along with providing materials.
    - **Attributes:**
        - instructorID: The ID of the instructor.
        - instructorName: Name of the instructor taking the course.
        - emailID: ID of the instructor.
        - password: Password set by the instructor to login.
    - **Operations:**
        - login(): To login.
        - resetPassword(): To reset the password if the instructor forgets.
        - scheduleCourses(): To schedule the course.
        - cancelCourse(): To cancel the course scheduled by the instructor.
        - updateCourse(): To update the information of the course.
        - trackProgress(): To track the progress of the learner.
        - gradingAssignments(): To grade the assignment for the learner.
        - provideMaterials(): Instructor provided materials to learners like a ppt, reference books, reference links, video clips.
        - assignTasks(): Instructors assigns tasks such as quizes, assignments, video learning, hackathon, etc.
    - **Relationships:**
        - A instructor can have one or many learners. Instructor offers one or many courses. Instructor can create one or many tasks.
2. **Courses:** A learner is capable to take the course and the instructor offers the course.
    - **Attributes:**
        - courseName: Name of the course.
        - courseID: Id of the course.
        - modules: Number of modules listed in the course.
        - description: Description of the course.
        - prerequisite: Prerequisite for the course.
    - **Relationships:**
        - A course of offered by one or more instructor. A course produces one schedule. A course is enrolled by zero or more learners.
3. **Learner:** Learner is capable of taking the courses, viewing, checking for the payment status, can download the study materials and achievement report. Learner can also fill up the feedback form for

the course and for the instructor as well.

- **Attributes:**
  - learnerID: Id of the learner.
  - name: Name of the learner.
  - email: Email of the learner.
  - password: Password of the learner to login.
  - learnerStatus: Status of the learner whether active or not.
- **Operations:**
  - learnerLogin(): Learner logins to the system.
  - recoverPassword(): Learners can recover the password if they forgets.
  - viewCourse(): Shows the courses the learner is currently taking.
  - addCourse(): Learner can enrol to the course.
  - checkEnrollmentStatus(): Shows the status of the learner enrolment.
  - checkPaymentStatus(): Learner can check the payment status.
  - downloadMaterials(): Downloads copies of any documents uploaded in the course by the instructor.
  - achievementReport(): Learner gets the achievement report which includes their leaderboard position, award badges and certificates.
  - courseFeedback(): Learner can provide the feedback on the course.
  - instructorFeedback(): Provision of feedback on the instructor.
- **Relationships:**
  - A learner gets a schedule. A learner enrols zero to many courses. A learner has one instructor. An instructor has one to many learners. A learner makes a payment, and a learner is assigned with a grade.

4. **Tasks:** Learner is assigned with the tasks like assignments, quizzes, exams along with deadlines and possible grade points that the learner can achieve.
   - **Attributes:**
     - taskName: Title of the task and this could be in form of words such as ‘’Assignment’, ‘Quizzes’, ‘Exams’, etc.
     - taskno: A reference number for a given task.
     - gradePoints: Maximum grade point possible for the task.
     - deadline: The time the task is supposed to be handed in.
     - materials: The material could be ppt, book, reference links or videos, etc.
   - **Relationships:**
     - Each task provides one grade, and each task is assigned by an instructor.
5. **Grades:** Learner gets the grade points along with feedback and recognition from the instructor for each assigned task.
   - **Attributes:**
     - learnerName: The identification of the learner.
     - gradePoints: Points earned by the learner in a specific course.
     - awardBadges: Recognition received by the learner for his/her performance.
   - **Operations:**
     - provideFeedback(): Enables an instructor to give feedback on either an assignment or a quiz.
   - **Relationships:**
     - A grade is given to one or many tasks and a grade is assigned to one or many learners.
6. **Administration:** Responsible to create the report, provide alerts, gets payment status and handles security restrictions.
   - **Attributes:**
     - adminID: Administrator unique ID.
     - adminName: Name of the admin.

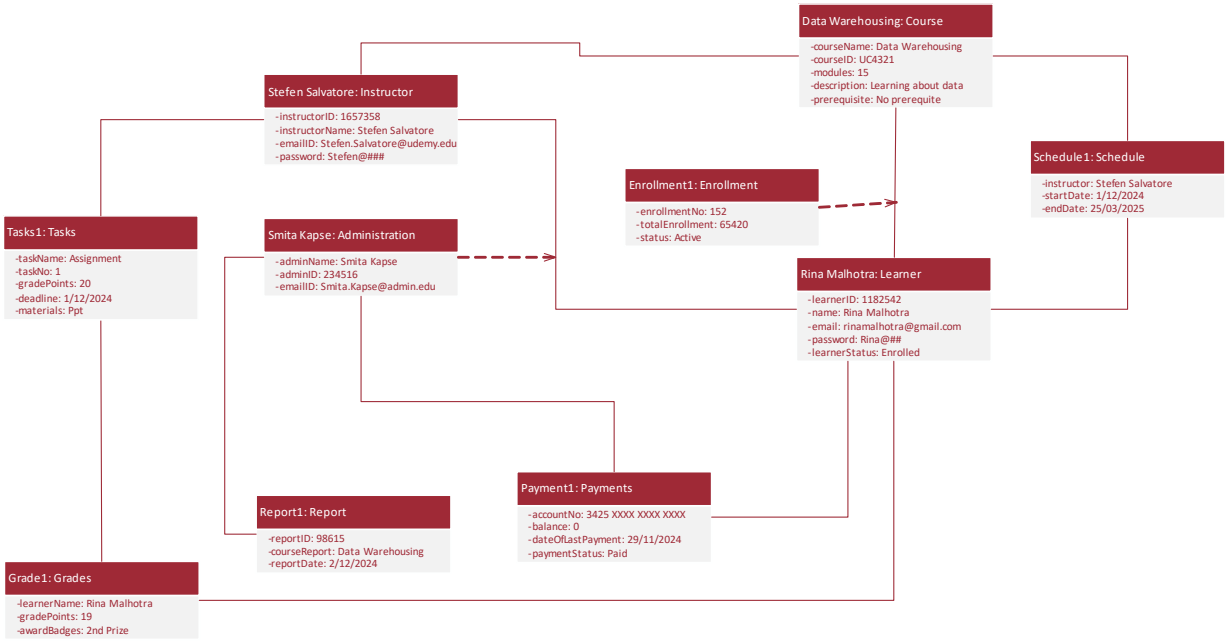
    - emailID: The administrator's email address.
- **Operations:**
    - createReport(): Prepares reports to track the number of learners enrolled in the courses.
    - provideAlerts(): Provides notification to the users if the system is facing any downtime. Provides notifications on upcoming assignments.
    - accessRestriction(): Provides access to specified users only.
    - getPaymentStatus(): Gets the payment status from the payment department for the learner.
- **Relationships:**
    - An administration generates zero or many reports. An admin receives one or more payments. An admin is also an association class of instructor and learner. Admin class provides additional details to the instructor and learner class.

7. **Report:** For documenting number of learners enrolled in the course. To understand how the course is going.
    - **Attributes:**
        - reportID: Document number that helps in identification of the report only.
        - coursereport: Name of the course for which the report is generated.
        - reportDate: The date the report when it was prepared.
    - **Operations:**
        - learnersEnrolled(): To get the total learners enrolled in a particular course.
    - **Relationships:**
        - Zero or many reports are generated by an admin.
8. **Enrolment:** An association class. It provides additional details of learner and course. It updates number of enrolment count.
    - **Attributes:**
        - enrolmentNo: The number to identify a record of enrolment.
        - totalEnrolment: Total head count of learners in each course.
        - status: The state of the learner, for example, active or inactive.
    - **Operations:**
        - updateTotalEnrolment(): Modifies the total number of enrolments for a course.
    - **Relationships:**
        - Association class between course and learner. This class provides an additional information to other classes.
9. **Payments:** It handles the payments for the learners. And updates the status of the payment as paid and not paid to the administrator.
    - **Attributes:**
        - accountNo: The account details of the learner.
        - balance: The balance of the learner.
        - dateOfLastPayment: Last payment made by the learner.
        - paymentStatus: The status of the payment; whether it was paid or is pending.
    - **Operations:**
        - updateBalance(): Updates the total balance for the learner after the payment is done.
        - updatePaymentStatus(): Updates the payment status from paid to not paid.
    - **Relationships:**
        - A payment is received by an admin department. A payment is made by one or many learners.
10. **Schedule:** To provide course schedule to the learners.
    - **Attributes:**
        - instructorID: The instructor ID who is in charge of the schedule.
        - startDate: The date on which the course or the schedule begins.

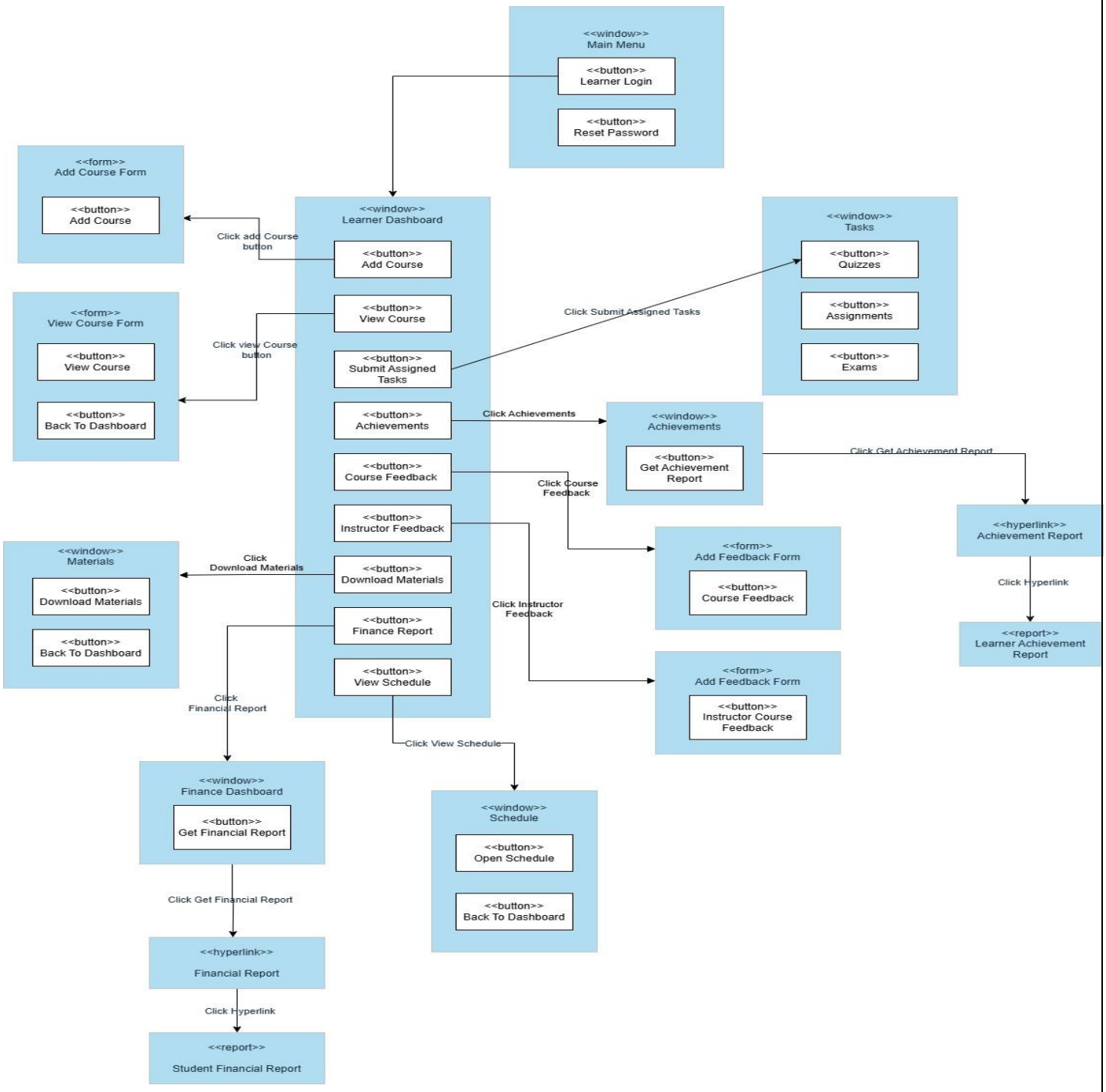
- endDate: The finish date the date on which the course or schedule is completed.

- Operations:
  - viewSchedule(): Learner can get to view their schedule
- **Relationships:**
  - One schedule is received by a learner. A schedule is produced by one or more course.

Object Diagram:



Windows Navigation Diagram: From learner`s perspective.



Conclusions:

A thorough overview of the Online Learning Management System, emphasizing its essential elements, features, and user interactions, has been given by this analysis. The structure and behavior of the system are clearly depicted in the class diagram and use case diagram. This project will help the learning mangement work smoothly. However, few technical things needs to be considered. This LMS can be modified in future to provide the corporate employees with employer specific training. This is the future scope of this project.

Technical Recommendations:

1. **Safety:**
    - **Data encryption:** To safeguard private user information, deploy robust encryption techniques. Use strong authentication techniques, such as multi-factor authentication (MFA), for secure authentication.
    - **Frequent security audits:** To find and fix vulnerabilities, conduct regular security assessments.
    - **Access restrictions**: To prevent unwanted access to data, use granular access controls.
    - **Compliance:** Comply with applicable data privacy laws (such as the CCPA and GDPR).
2. **Scalability:**
    - **Scalable infrastructure:** Verify that the system can manage rising traffic and storage needs.
    - **Load testing:** Ensure that the system is not lagging. Run load tests frequently.
3. **Performance:**
    - **Frequent performance monitoring:** Keep an eye on system performance and spot possible problems.