Project Overview

Form a team of 4 students to design and develop a learning platform integrated with GenAI to facilitate teachers to deliver student-centered, personalized, and interactive learning in the age of Education 4.0. Some high-level ideas/requirements will be discussed during the lecture.

To determine what to build:

- Understand the needs and pain points of stakeholders (e.g. teachers)
    - Discussion in social media (YouTube, Reddit, etc)
    - Interview the major stakeholders (e.g. teachers) to understand their pinpoints and needs
- Understand the features and problems of the existing platforms and identify improvement opportunities
- Demonstrate some uniqueness, interesting/innovative elements in the project
    - Not another AI-generated app or apps with just the general features (e.g., QA using RAG)

You should study the existing platform, its limitations, and explore how your team may develop a platform with improved/more innovative features and better user experience. Each group may create multiple prototypes for idea brainstorming/validation, but should collaboratively work on one final app with integrated features

- Each team member should demonstrate sufficient contribution to the project (e.g., develop different features and integrate with the final app)
- Good variety of features with sufficient complexity for different team members to collaboratively and meaningfully develop the app
- Collect feedback using the prototypes from the stakeholders for iterative improvement.

In the project, you are welcome to use any programming languages, frameworks, and libraries, etc. You can also use GenAI for your project (e.g., idea brainstorming, designing, evaluating alternatives, development and coding, testing, etc).

- Try to understand how AI is implementing the solution, and learn new skills along the project
- Demonstrate human judgement and critical thinking
- Remove the “AI smells” from your app and the submitted deliverables (e.g., Report)

Group Formation

Students should form a group of 4 members by completing the MS form by the end of week 3 (i.e 31/1/2024 (SAT)). Students without a group will be randomly assigned a group after the deadline.

The GitHub classroom invitation link will be available soon. The team leader must accept the invitation and create the team's project repository in our course project's GitHub organization. The repository should be private and accessible only to team members and the instructor. The team leader will be the repository admin and should invite the other team members to collaborate. During the project,

- Each team member should use the GitHub repository (e.g. issues, project board, etc) to manage and collaborate on your project.
- Your project source code and other related files (e.g. documentation) should be uploaded to your project repository in the course project's GitHub organization.
- All students in the group should actively participate during the software development (E.g., coding, testing, writing documentation) and adopt good software engineering practice

The project meeting will be held during the lectures in week 8/9. Each group will have around 5-10 minutes to discuss its project progress. You should prepare a 1–2-page summary of your progress, division of work, and your system design/prototype for discussion. We may also clarify the requirements in the meeting.

Week 11: Final Submission

Deliverables

- The prototype should be deployed to some publicly accessible platform (e.g. Vercel)
- A 5–8-minute overview video of your system and how to use the platform
- Your presentation slides and a final report
    - Overview of your project, system design and features
    - Discuss your software development process (requirement, design, development, testing, deployment, etc) and how your team collaborates in your project
    - User manual for admin, instructor and students and links to demo videos of the features

Week 12: Presentation

The presentation will be held in week 12 and week 13. Each group will have around 8-10 minutes (to be confirmed) to present their project. The presentation slides should cover the following aspects:

- An overview of your project, features, and system architecture.
- The demo and features of your prototype/system (You may show your demo video)
- Sharing findings and reflection on the project. Examples:
    - Development Process
        - Briefly introduce your team's technical/programming background and discuss your development process and examples of how your group collaborates. Discuss how each team member is contributing to the project and the challenges encountered (if any)
        - How do you design the UI? How did you improve the UI and UX of the platform?
        - Which feature in the project was the most challenging to implement, and how did you solve the problem?
    - Use of GenAI in software development
        - Which parts of the code are generated by AI and which parts are manually coded? How is your team collaborating while using GenAI to complete the software development tasks? What works well/not so well?
        - Have you used GenAI to support the software testing and perform other tasks in software development? Have you done any refactoring? What are the good practices when using GenAI for software development? Give examples to showcase how your team are using GenAI to assist in your project development and implementation.
    - Overall reflection: Reflect on what you have learnt/the findings from the project, what you have done well/not done so well, what the challenges and solution, etc.
- Appendix: You may include other information/screenshots that are relevant to your project.

Evaluation Criteria

- Project discussion meeting (5%)
- Final presentation and deliverables (85%)
    - System features and prototype (40%):
    - Presentation, Sharing, findings, and reflection (45%)
- Peer evaluation (10%)
    - Each group should
        - Try the platforms of two other groups assigned to you
        - Attend the presentation of your two assigned teams and ask at least one question
        - Evaluate your peers' platform/presentation and provide feedback.

Rubrics (Tentative)

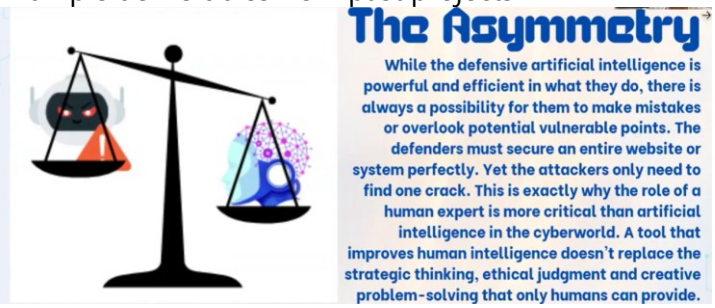
Criteria	Good	Satisfactory	Needs Improvement
Project Discussion Meeting (5%)	Clear progress summary with a well-defined prototype and division of work. Team shows active participation and a strong grasp of requirements.	Progress shared but limited details on the prototype or task division. Participation is evident but uneven among members.	Lacks clarity on progress or work division. Prototype/design unclear. Some members appear disengaged.
System Features & Prototype (40%)	Features are **innovative/tailored** to the needs of the stakeholders. Clear articulation of motivations, differentiation from existing tools. **GenAI features** are well integrated. Prototype is functional, responsive, and deployed with good UI/UX.	Features are implemented but reflect more **basic or imitative functions**. Motivation and differentiation are partly explained. Prototype deployed but with limited interactivity or UI/UX polish.	Features are very **basic or similar** to existing tools. Lack of motivation or justification. Prototype incomplete, not responsive, or not deployed.
Presentation, Sharing, and reflection (45%)	**Clear, engaging, professional delivery** with confident pronunciation, varied tone, and smooth demo. The slides are well-structured, with clear visuals that enhance understanding. Provides **deep reflections**: a clear discussion of the development process, team collaboration, and GenAI usage (successes & challenges). Insights are thoughtful and specific. Clear takeaways about the learning process.	Presentation is understandable but less polished. Delivery may include minor clarity/pronunciation issues; slides/content may be somewhat repetitive or cluttered. Audience engagement is limited. Reflections are present but **general or brief**. The development process and GenAI usage are described superficially. Limited insights about design choices, challenges, or lessons learned.	Presentation **unclear, monotonous, or poorly structured**. Slides confusing with missing sections. Demo ineffective or absent. Audience disengaged. Reflection **minimal or missing**. No real insights beyond surface-level description. Lacks discussion of GenAI usage, challenges, or lessons learned.

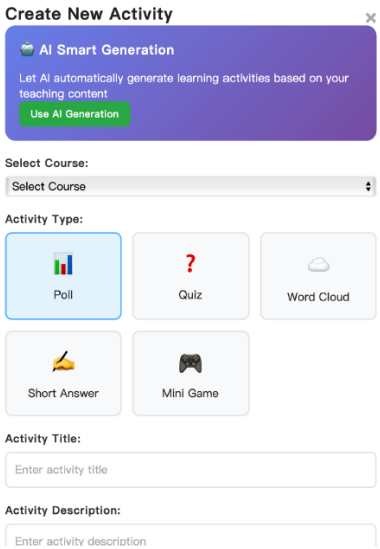
Appendix

Do you like the presentation shown in this video? Why?

https://youtu.be/F2FmTdLtb_4?si=StQAnbl27e_J1Djx

Example deliverables from past projects





- **Navigation Authentication**
  - **Protected Routes**: Authentication required for most application routes
  - **Public Routes**: `/login`, `/change-password`, `/join-activity/*`, `/activity-login/*`
  - **Role-Based Access**: Different navigation patterns for admin, teacher, and student roles
  - **Token Validation**: Automatic token verification and user info restoration

Parameter Passing Mechanisms

- URL Query Parameters (`route.query`)
  - **Primary Method**: Main mechanism for passing complex data between pages
  - **Data Types**: Supports string values that can be parsed into complex objects
  - **Persistence**: Parameters persist in browser URL and can be bookmarked
  - **Access Pattern**: `route.query.parameterName` for retrieval
- Route Parameters (`route.params`)
  - **Usage**: Path-based parameters for resource identification
  - **Pattern**: `/resource/:id` format in route definitions
  - **Access**: `route.params.id` for retrieval
  - **Type Safety**: Automatic TypeScript type inference
- JSON String Parameter Encoding
  - **Complex Data**: Activity configurations passed as JSON strings

2.4.7 Design Trade-offs

2.4.7.1 Data Redundancy vs. Performance

Redundant design is adopted in the following scenarios.

- course.teacher_name: Avoid frequent JOIN operations on the user table
- Multiple *_name fields in assign_submit: Improve the performance of homework list queries by 50-70%

Trade-off: Adopt redundancy in scenarios with more reads and fewer writes; maintain normalization for frequently changing data.

2.4.7.2 Flexibility vs. Constraints

- **Flexible Design**: Store dynamic content in JSONB, reserve extension fields, and store time in Text type
- **Constraint Design**: Unique username, non-null constraint, primary key constraint

Decision: Strong constraints on core fields and flexible design for content fields.

2.4.8 Security Assurance

- **Password Security**: Encrypted with hash algorithm, stored irreversibly
- **Access Control**: Role-based permission control, where teachers can only access their own courses
- **SQL Injection Protection**: ORM parameterized queries, Pydantic data validatio

Executive Summary

This report covers the complete integration test suite for admin functionality, containing 133 test cases across 4 major functional modules. All tests passed successfully, achieving 100% functional coverage. Integration tests verify complete business workflows including database operations, external API calls, user management, course management, and Moodle synchronization scenarios.

Module Scope

- ✅ **User Management**: 43 tests
- ✅ **System Settings**: 16 tests
- ✅ **Course Management**: 49 tests
- ✅ **Moodle Integration**: 25 tests

Overall Test Results

Module	Tests	Status	Duration	Complexity
User Management	43	✅ 100%	~1.75s	High
System Settings	16	✅ 100%	~0.86s	Medium
Course Management	49	✅ 100%	~2.0s	High
Moodle Integration	25	✅ 100%	~0.89s	Very High
TOTAL	**133**	✅ **100%**	**~5.5s**	-

Success Metrics

- ✅ **133/133 tests passed** (100% success rate)
- ✅ **100% coverage** for all integration scenarios
- ✅ **Zero failures** across all test runs
- ✅ **Complete workflow testing** (database + external APIs + email)
- ✅ **All edge cases and error scenarios** covered

Video: https://www.youtube.com/watch?v=xxqKrJtJozM