Project Requirements Document - version 1

Authors:

- Xichen He
- Kyrie Cai
- Ning Xia
- Bowen Duanmu
- An Cao

Team name: GoCSIL

Company Name: Algorand Foundation

Project title:

AlgoLearn Learning Management System

Background and Motivation:

The Learning Management System (LMS) is crucial in our college learning process, as a system that sustains most of the studying materials, records our grades and performances, and allows course instructors to administer the students. In a world where the internet has long become irreplaceable, such a system provides great convenience to integrate all the information within the scope of a course, and output useful functionalities to both students and instructors.

Not to mention, universities have already come up with their own LMS that includes basic features listed above for the need of managing classes and distributing digital materials in a “centralized” manner. In most cases, the course instructor defines the form of usage of the system, such as what components will appear on a course’s web page and how students might utilize them. While this provides convenience for the class, there is room for improvement. If we shift our focus to students, and especially their learning experience, we can see that current LMSs are very instructor-centered and students are only playing the role of attendee in the system, merely receiving instructors’ direction of the course and submitting what instructors require, but this betrays the true atmosphere of a classroom. We want to enhance the role of students’ experience in the LMS, let their voices be heard, and make their voices be heard, their actions be counted, in the process of their own learning.

Goals：

Algorand has been noticing the idea of Next-Generation Digital Learning Environment (NGDLE), which is leading the rapidly changing pedagogical landscape as it sees engagement, collaboration, mobility and personalization as main drivers in the future of LMS. We narrowed

down the aim of NGDLE to the scope of our capstone project, where we will seek to build a system emphasizing realization of fundamental features, including tracking attendance, enhancing engagement and enriching the students-professor experience, via the form of course-specific NFT transactions. We aim to deploy designs to gamify the NFT-related processes, make such practices capable of restoring more perspectives of the experience of "learning in the course" beyond attendance or homework and exam grades, and put it in a Web 3 blockchain where credibility and transparency of data are more guaranteed. We will still ensure and most likely amplify the convenience for the instructor side to manage such a system, and allow students' attitudes to easily come through for the instruction team over the system, creating possibilities for whom to timely adjust upon feedback.

Implementation choices:

For the web application, we are using beaker for the framework because it has a package for generating a client to interact with the app and we have the link and case example. For the frontend, we are using React JS and it will bring more flexibility to the UI design. We are choosing MongDB as our database since it's a non-relational database that works pretty well with the NFT trading and store. The database will be used to store linked emails with Algo accounts, badges/quests exchanged in the form of NFT and user profile information. From these, we can easily get the performance of students and their behaviors are more transparent, which gives the possibility to grade a student from different aspects, instead of only grade matters. For smart contracts, we will use Python >= 3.10, PyTeal, Beaker. For local Algod to test/dev, we will Docker, Sandbox. Also, for the authentication, we will use the ARC code provided by our mentor.

Core Components:

- **Attendance**: This system allows professors and students to verifiably prove attendance. Professors can easily create attendance badges for that class and that day in the form of NFTs. Students can scan the QR code issued by the professor to opt in the asset and receive the badge. Professors can see the overall class attendance and students can only see personal attendance.

- **Badges**: This component enables professors to issue badges upon various event. Professors can create classes of badges to reward or punish students. For example, best question of the day, active in class, late to class, etc.

- **Groups**: This component enables professors to create groups for class and assign students within class to groups. For students, groups allow them to communicate with each other. For professors, groups keep track and store proofs of interactions and achievements. Professors can issue badges to groups based on their performance, for example, best presentation.

- **Forums**: Both professors and students can post, discuss, and up/downvote topics. It allows students to ask and discuss class materials for better understanding. This can be used by professors to track and reward most engaged students via issuing badges.

Assumptions:

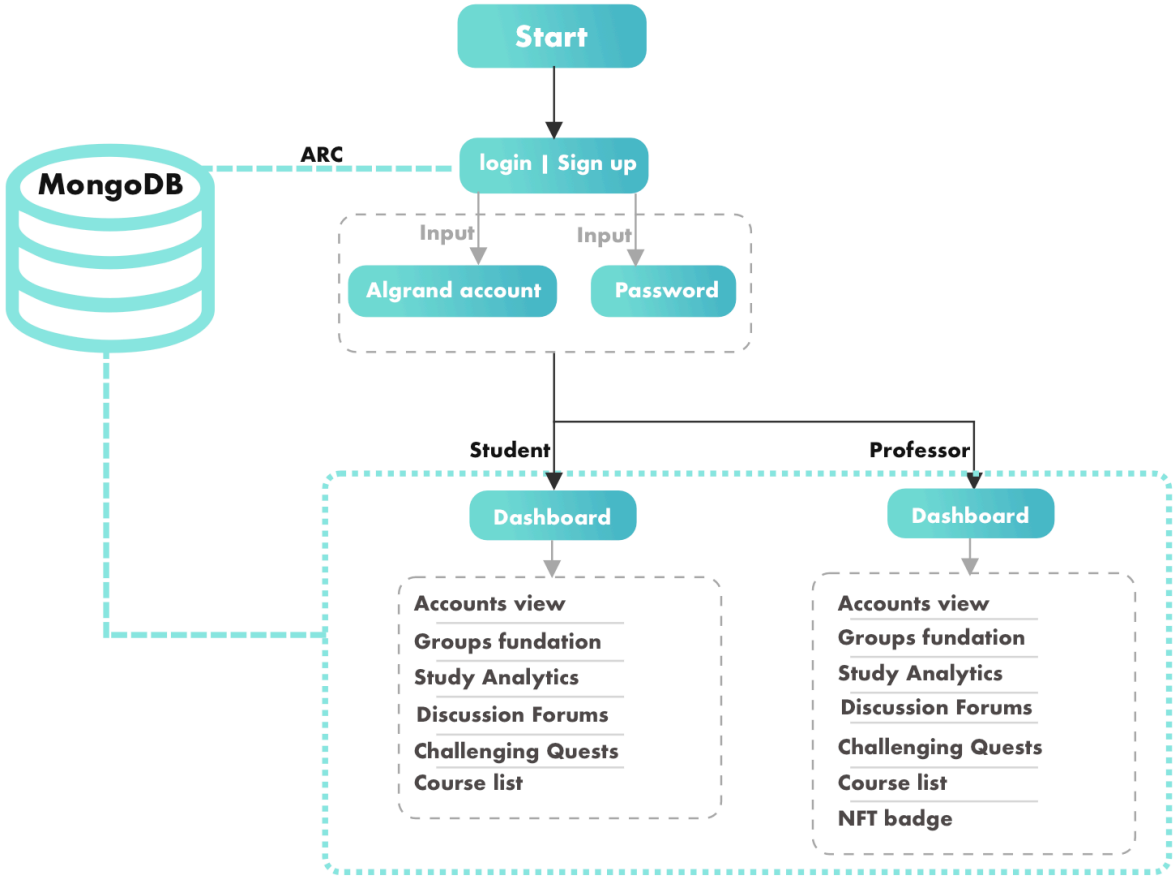
- Our target user will be professors and students
- Users will need to log in to access and manipulate data
- Users are able to create Algorand account

Stretch Goals:

We may include additional features that allow for more personalized customization in configuration of the dashboard to allow users to choose their favorite alignment and functionality.

We may also include analytical features that can present a report of the student's study data and related study suggestions to help improve course behavior.The report will include generalized data about the attendance,homework grades,number of class activities participation(such as question posting and answering, meeting with professor) and NFT changes. The student can also see the past reports. The report will evaluate the student behavior based on such data and identify the improvements and weaknesses in the student study behavior. After the report, a list of suggestions will be given based on the above aspects.

Architecture diagram:



Users stories:

<u>General User requirements</u>

As a new user, I want to see a clean looking landing page, so I have an easier time navigating through the site.

- Acceptance Criteria: The website has clearly labeled tabs for users to create or sign in to an account.

As a new user, I want to be able to sign up for an account if I do not have one already.

- Acceptance Criteria: The user has access to registration form via sign up button on the page. Once the user fills out the form, their login information will be stored so that they can login.

As a user, I want to log in my account that is associated with an Algorand NFT wallet.

- Acceptance Criteria: The user has access to log in form and is able to enter their username and password to login. After logging in, they should be redirected to the account page
- Scenario 1: The User types in correct credentials and successfully logins to the website.
- Scenario 2: The User types in incorrect credentials and an error message pops up.

As a user, if I forget my account password, I am able to reset my password with my email.

- Acceptance Criteria: The user can submit their email address to receive password reset email. If their email is registered, a password reset link will be sent. Otherwise, it will pop out a message of an unregistered email.
- Scenario 1: If the email put in is registered, a password reset link will be sent to their email.
- Scenario 2: If the email put in is not registered, it will pop out a message to notify the user that the email is not registered and reset link will not be sent.

As a user, I can see a clean and intuitive dashboard

- Acceptance Criteria: After login, the dashboard should have dropdown menus and navigate to the corresponding page.
- Scenario 1: Users can click each module on the dashboard to open the dropdown menu and see what it has.
- Scenario 2: Users can click each part of the dropdown menu to navigate the corresponding page.

<u>Student Users specification</u>

As a student user, I can view my NFT and profile on the main page of my account.

- Acceptance criteria: The NFT and profile is accessible on the main page
- Scenario 1: The main information page display is successful
    - Given a user trying to access his main page
    - Give a overview of the NFT and profile information to users

    - The user can view each detail in the profile
- Scenario 2: The main information page display can not be accessed
    - Given a user trying to access his main page
    - The user could not see all the information or enter the main page
    - The user will see a reloading or sign in button.

As a student, I want to scan a QR code issued by the professor and receive NFT.

- Acceptance Criteria: The QR code can be successfully scanned and students will receive the correct attendance NFT for that class that day.

As a student, I want to see my class performance.

- Acceptance Criteria: The student can see the personal attendance for each class correctly, including the overall attendance ratio and attendance on each class day.

<u>Professor Users specification</u>

As a Professor, I can publish and mint new NFT to the class

- Acceptance Criteria: The NFT can be correctly created and published to the software
- Scenario 1:The professor successfully published his NFT and make it accessible for student in his class
- Scenario 2:The professor can not badge his NFT and an error would occur

As a professor, I want to create an QR code for class and track attendance.

- Acceptance Criteria: The unique QR code can be successfully created for that class that day and presented to the professor.

As a professor, I want to check the attendance performance for each class.

- Acceptance Criteria: The professor can see the overall attendance for the class and also the attendance for each student.

As a professor, I want to create groups for classes and for teams within class.

- Acceptance Criteria: The professor can create groups for class and assign students within class to groups. Groups allow students to post messages.

Appendix:

Technology:

- **For smart contracts:** Python >= 3.10, PyTeal, Beaker
- **For local Algod to test/dev:** Docker, Sandbox
- **For front end:** React JS
- **For database:** MongoDB
- **For framework:** beaker package for clients to interact with the app, React JS