Homework #2

Course Management System

As the project description states, the course management system is an online management software application designed for educational institutions. The primary goal of the project is to facilitate seamless interaction between students and instructors in schools, colleges, and universities concerning the submission of projects, assignments, theses, and receiving feedback from instructors.

The functionality will be provided as a web application running a MERN stack:

- ➔ React will be used to render the frontend and provide functionality to all the components.
- ➔ Express will be used to handle backend tasks, logic, and as a middleman for CRUD operations between client and server.
- ➔ MongoDB will be used as the database containing and handling information needed to reliably store information related to each use case.
- ➔ NodeJS will be used as the backend that scaffolds the rest of the stack as well as integrates modules before compilation.

The three modules are the administrator module, student module, and instructor module.

Top-level use case examples are:

Instructor:

- *Log in*
  Functionality for instructor login will be provided in one of two ways:
  (1) A list of instructors/faculty will be scraped from the auburn.edu, given elevated permissions within the database, and logged in via simple HTTP basic auth with a randomly generated password sent to their email.
  OR (2) A dummy login portal will be set up where users that have been added to the database by administrators can simply provide their username and log in with no authentication. The choice of implementation for this case depends on complexity, case (2) will be implemented first, and then refactored into case (1) if time constraints allow us to do so.
- *Add coursework/ course information*
  Instructors need to be able to upload, schedule, assign, and remove coursework and syllabi. Two ways to provide this are (1), our CMS service will allow local file upload with a maximum file size and with a heavily constrained format list, the files can then be downloaded by the students when assigned (in-browser viewing not planned to avoid third party library use). Alternatively (2), we may choose to allow a simple text entry as the only assignment submission type to reduce complexity. Again, we will implement case (2) and refactor into case (1) if there are no time constraints.
- *View and 'grade' submissions*
  Instructors need to be able to view student submissions in the same manner. This goes

along with the above case. If file upload is allowed, then the instructor can download the submission and provide a grade from a numerical entry. The grade will then be assigned to the student in the database and can be viewed when the student logs back in.

- *Edit coursework/course information*
  Instructors will be able to arbitrarily modify parts of their course, for example, the dates the course is active, the students admitted into the course, the modules that are available to the students, and the syllabus for the course.
- *Display course*
  Instructors reserve the right to change visibility on their courses by enabling, disabling or removing them, even if students are enrolled in them. In this case, the course will show as disabled from their end. If a student is enrolled in a course that has not yet begun, the course will accordingly show 'not started'.
- *Edit profile*
  Instructors can edit their own personal profile and can modify their names, their avatars, and their office hours along with the meeting format. Their email is shown, but cannot be modified.
- *Add comment to submission/announcement*
  Also pertinent to the *view and grade submissions* case, instructors can also provide feedback in the form of comments that will be shown to the students.

Student:

- *Log in*
  The functionality for student login will work similarly to the instructor login, using the same database and login portal. As with the instructor login, student login will be provided one of two ways:
  (1) A list of students are any Auburn student with an @auburn.edu email that is not already registered as an instructor or administrator. They will log in via simple HTTP basic auth with a randomly assigned password sent to their email upon 'registration'. They will be given limited permissions after login.
  OR (2) A dummy login portal will be set up where students that have been added to the database by administrators or instructors can simply provide their username and log in with no authentication. The choice of implementation for this case depends on complexity, case (2) will be implemented first, and then refactored into case (1) if time constraints allow us to do so.
- *View coursework/course information*
  Students can login to view their assignments for each course, and they can navigate the course to view a syllabus, modules, instructor information, and their grades for the course.
- *Submit assignment*
  Depending on which implementation we use for course submissions, a student will either submit a text-based assignment, or upload a file. The file will not be viewable online to restrict reliance on external libraries, but can be redownloaded and viewed on the client machine by both the instructor and the student.

- *Edit profile*
  Students can change their name, their profile photo, and can add a bio for any prospective companies, in a similar vein to what Piazza does.
- *Add comment to submission/announcement*
  Students will be able to submit a supplementary comment to any submission or announcement via a simple text entry. They can also comment on their assigned grade.

Administrator:

- *Manage employees*
  Administrators encompass any school user that must have control over the management and organization of the CMS. They have all functionality that instructors have, plus the ability to assign permissions to arbitrary users (TAs, for example), assigning instructors to departments where courses wait to be filled, as well as adding and removing instructors (hiring and termination cases)
- *Create curriculum*
  Administrators can create curriculums based on demand from incoming students and assign existing instructors within a department if their schedule does not conflict. They can define curriculum paths and requirements, class names and numbers, and set restrictions based on previous course requirements.
- *Remove course*
  While instructors can disable and ‘remove’ courses, they remain eligible for enrollment through other professors, as they are only withdrawing themselves. Administrators can fully remove courses from the curriculum.
- *Create course*
  As defined above, courses can be defined which have a certain structure, are offered at certain times of the year, can be taught by certain instructors, and are offered to students after a certain year or education level (Freshman -> senior or undergrad vs grad)
- *Modify course*
  Administrators can change any of the course requirements. For example, class capacity, number of offered courses, schedules for each course, instructors assigned per course taught, credit hours, et cetera.
- *Assign course to instructors*
  Administrators can override a temporary vacuum due to no longer having an instructor for a course by backfilling an instructor or a substitute teacher from a different department. For the remainder of the course, the database will show them as having their original qualifications, but will be flagged as temporary substitutes within the database.

Ideally, we will narrow down these cases to their bare essentials and work using an iterative approach, building upon previous work, to ensure that we implement as much of the functionality as possible without underdelivering. The goal is to implement simple base cases that meet minimum requirements and expand as time and effort allows. We will largely use Canvas LMS as a reference point.