Team name: We_cant_think_of_a_name

Current sprint User Stories are in yellow

Issue #	Task Title	User Story	Story Points (out of 5)	Description / Associated Tasks	Due Date	Priority Level (LOW, MEDIUM, HIGH)	Risk Level (LOW, MEDIUM, HIGH)	Responsibles	Sprint
#8	Setup database for professor credentials	"As an instructor, I want my email and password to be stored securely"	4	- Set up *AtlasDB* so that database is stored online via *Mongoose* - Add proof of concept code that can upload new users to database (with encryption) - Test everything on a terminal before linking to rest of the project	2024/09/26	HIGH	HIGH	Robert & Othmane	1
#54	Backend and Frontend merge in dev		1	On *GitHub*		MEDIUM	LOW	Hendrik	1
#11	Implement authentication procedure	"As a user, I want to log in with an extra sense of security"	3	- Validate student credentials (id & password) against database - Validate instructor credentials (email & password) against database. - Ensure that students are redirected to their specific dashboard upon successful login - Ensure that students are redirected to their specific dashboard upon successful login - Use encryption to store and compare passwords securely			HIGH	Robert & Jana & Yousef	1
#9	Design the student login page UI	"As a student, I want a clean, clear and visually appealing webstte where I will be able to log in to access the peer reviewing platform"	2	- HTML to create and place elements and form - CSS to add themes and colors and aesthetics - Javascript and React to make the page responsive - Ensure form is responsive	2024/09/26	HIGH	LOW	Elif & Jana	1
#12	Create session management for logged in users		3	- Implement sessions/tokens for student authentication - Implement sessions/tokens for instructor authentication	2024/09/23	HIGH	LOW	Jana & Yousef	3
#6	Set up project		1	- Setting up tech stack - Set up *gitignore* - Set up file system & folder structure - Create *README.md* file - Set up issue tracking - Test repository set up	2024/09/12	HIGH	HIGH	Yousef & Elif	1
#7	Setup database for student credentials	"As a student, I want my email and password to be stored securely"	4	- Set up AtlasDB so that database is stored online via mongoose - Add proof of concept code that can upload new users to database (with encryption) - Test everything on a terminal before linking to rest of the project	2024/09/26	HIGH	HIGH	Robert & Othmane	1
#13	Invalid hook call error when adding React router		2	Issue faced: when adding router setup, app breaks and there is an error on F12 Console (*warning: Invalid hook call. Hooks can only be called inside the body of a function component.*) Help wanted.		LOW	LOW	Yousef & Hendrik & Jana	1
#19	Design instructor login page UI	"As an instructor, I want a clean, clear and visually appealing webstte where I will be able to log in to access the peer reviewing platform"	2	- HTML to create and place elements and form - CSS to add themes and colors and aesthetics - Javascript and React to make the page responsive - Ensure form is responsive	2024/09/26	HIGH	LOW	Elif & Jana	1
#20	Design student dashboard UI	"As a student, I want ot have an ovreview of the website on a single page so I can navigate where I wish"	2	- Create and place elements using HTML - Each course the student is taknig that requires peer reviewing appears on the website - Create API to get student data. - Map all student data to courses/teams they are part of		HIGH	LOW	Elif & Jana	2
#21	Design instructor dashboard UI	"As an instructor, I want ot have an ovreview of the website on a single page so I can navigate where I wish"	2	- HTML, CSS, Javascript, React - Implement sidebar - Overview of courses - Create new routes for API: creating course, adding students to course - Create model schemas for courses and teams - Add new functions to interact with database-API: create course and add student to course		HIGH	LOW	Elif & Jana	2
#24	Link website - server - database		2	There's a problem merging the branches, possibly because of our unfamiliarity with using MongoDB. Some changes and files don't appear after merge.		HIGH	MEDIUM	Robert & Youssef & Hendrik	1
#27	Merge what was done so far to main branch		2	On GitHub	2024/09/30	HIGH	LOW	Hendrik	1
#28	.env file to be removed from gitignore?			On GitHub				Hendrik	

#29	Team Management System	"As an instructor, I want to created the teams for the peer assessment manually"	4	- Create several new routes for the API: for getting the courses (instructor), for getting the students (instructor), for adding students to team (instructor) and for creating teams (instructor) and for displaying the existing teams (instructor). - GET /instructor/courses: Retrieve all courses that the instructor is teaching. @hendriktebeng - GET /instructor/courses/:courseId/students: Retrieve all students enrolled in a specific course. - POST /instructor/create-team: Create a new team by providing the team name, course ID, and selected student IDs. - POST /instructor/add-student-to-team: Add a student to an existing team. - GET /instructor/courses/:courseId/teams: Retrieve all teams created under a specific course. - Create model schemas for courses and teams. 1. Create Schema for Course Model: Fields: id: Unique identifier, courseCode: String, courseName: String, students: Array of student IDs, teams: Array of team IDs 2. Create Schema for Team Model: Fields: id: Unique identifier, teamName: String, course: Course ID , members: Array of students - Add new functions to interact with database-API: 1. createTeamForCourse(courseId, teamName, memberIds) : allows an instructor to manually create a team by providing the course ID, team name, and student IDs. 2. getAllCoursesForInstructor(instructorId): This function retrieves the list of courses assigned to a particular instructor. 3. getTeamsForCourse(courseId): This function retrieves all teams associated with a specific course. 4. addStudentToTeam(teamId, studentId): This function adds a student to an existing team by team ID and student ID. 5. getAvailableStudentsForCourse(courseId): This function retrieves all students in a course, excluding those already assigned to a team. 6. createCourse(courseCode, courseName, instructorId): This function allows the instructor to create a new course and store it in the database. 7. addStudentToCourse(courseId, studentId): This function adds a student to a course by course ID and student ID. ### **Front-End:** - Create page to interact with API. Course Management: 1. Display a dropdown to select a course. 2. Provide a form to create a new course by entering the course code and course name. Student Management 1. Display a multi-select box for the instructor to add students to the selected course. 2. Use API integration to update course-student relationships and reflect them in the UI. Create a team: Display a section for team creation with: 1. A textbox to enter the team name. @jana-madhoun 2. A multi-select box for selecting students (filtered by those not already in a team). Add members to a team: 1. Provide a form to add students to existing teams with dropdowns to select teams and students.	2024/10/24	HIGH	MEDIUM	Hendrik, Robert & Jana	2
#30	Peer Assessment UI	"As a student, I want to have an interface that allows me to pick teammates for evaluation, so that I can easily provide feedback to my peers during assessments"	4	Front End: - Design and Implement Interface: UI that lists teammates in a table format that displays the Name, Email and an Assessment Link that redirects to the peer assessment form for that student. - Selection and Redirection Functionality: Ensure that clicking on “Assess” redirects to the peer assessment form for the selected student. - Dynamic List Update: Ensure the table displays the latest list of teammates for evaluation. Back End: - Add new functions to interact with database-API: to create teams (instructor), to get all courses (instructor) and to get all created teams (instructor). - Database Operations: Teammates, teams, and assessments are properly stored and fetched from the database. - getTeammate(teammateId): Retrieve a teammate's information. - getStudentTeammates(studentId): Fetch teammates for the logged-in student. - submitAssessment(evaluatorId, evaluateeId, assessment): Create a new peer assessment. - submitAssessments(evaluatorId, assessments): Submit multiple assessments for different teammates. API: Create the following routes: - GET /student-teammates returns a list of teammates with all required data. - POST /submit-assessment correctly saves individual assessments. - POST /login authenticates students and returns the appropriate user data. - GET /course-students/:courseId Retrieve students enrolled in a specific course by courseId.	2024/10/25	HIGH	MEDIUM	Elif, Robert & Hendrik	2

#36	7-Point Peer-Assessment Scale	"As a student, I want to provide peer ratings using a 7-point scale for multiple assessment criteria and submit my evaluation, so that I can offer feedback to my peers."	3	- Create model schemas for assessments. - Create database functions to store assessments - Create an invididual assessment page to allow students to submit peer evaluation	2024/10/26	HIGH	MEDIUM	Hendrik, Robert & Jana	2
#63	Detailed Assessment Summary	"As an instructor, I need a clear overview of student assessments for each course I teach so I can evaluate team performance efficiently."	3	Front-End Tasks - Fetch and Display Data - Implement `fetchDetailedData()` to retrieve detailed assessment data from the backend API. - If the response is successful, parse the data and set `teamsData`. - Handle errors by logging them to the console. - Display Team and Student scores and comments - For each team in `teamsData`, create a section displaying the team name and a list of students. - For each student: - Display their full name. - Create a table showing scores provided by each peer in Cooperation, Conceptual, Practical, Work Ethic, and the average score -Below each student's scores, display a comments section. API Tasks - Create a GET API Route: `/instructor/detailed-assessment` - Retrieve all teams from `teamModel`. - Populate each team with its associated course information (`courseCode`, `courseName`) and member details (`firstname`, `lastname`). - For each member: - Fetch all assessments where the student is the evaluatee. - Calculate Averages: - If no assessments exist, return `N/A` for averages. - If assessments exist, calculate average scores for each dimension (Cooperation, Conceptual, Practical, Work Ethic). - Compute the overall average as the average of all four dimensions. - Format assessments to include scores and comments for each dimension. - Construct and return a structured JSON response for each team, including each member's data and formatted assessments.	2024/11/07	HIGH	MEDIUM	Jana & Yousef	3
#64	Assessment Summary	"As an instructor, I want to view a summary of student assessments by course so that I can quickly understand student performance across key dimensions (Cooperation, Conceptual, Practical, Work Ethic) and track team contributions."	3	API/Database Tasks Front-End Tasks - Create Assessment Summary Page: - UI Elements: - Team Selector: Dropdown to choose a team to view. - Summary Table: Displays each team member with average ratings for cooperation, conceptual contribution, practical contribution, and work ethic. - Fetch and Display Data - Implement `fetchSummaryData()` to retrieve assessment summary data from the backend API. - If the response is successful, parse the data and set `summaryData` and `selectedCourse`. - Add Course Selection Dropdown - Create a dropdown that lists courses fetched from the backend, defaulting to the first course in the data if no course is selected. - On selection, update the `selectedCourse` state to reflect the chosen course. - Display Assessment Summary Table - Map `selectedCourseData.students` to display rows in a table. - Each row should include student ID, first name, last name, team, and averages for Cooperation, Conceptual, Practical, Work Ethic, and Overall, along with the number of evaluators.	2024/11/07	HIGH	MEDIUM	Elif & Hendrik	3

#65	Chat Feature for Peer Assessment System	"As a student or instructor I want a chat feature that allows me to communicate with my peers, instructors, and team members within a specific course, so that I can coordinate, discuss assignments, and clarify information effectively within my peer assessment system."	2	API Tasks - GET /chat/messages/: - Fields: - `courseId`: ID of the selected course. - Steps: - Validate `courseId` and check if the course exists. - Fetch all messages relevant to the course, including public, team, and private messages. - Sort messages by timestamp and return in the response. - Create a GET route to fetch chat recipients by course ID - GET /chat/recipients/: - Fields: - `courseId`: ID of the selected course. - Steps: - Fetch all potential chat recipients within the course context. - Include course instructor and students based on enrollment. - Return the list of recipients with basic user information. - Create a POST route to send a new message - POST /chat/send: - Fields: - `courseId`: ID of the selected course. - `senderId`: ID of the message sender. - `recipientId`: ID of the recipient for private messages. - `message`: Text of the message. - `senderType`: Type of the sender (student or instructor) - `teamId`: ID of the team (for team-specific messages). - Steps: - Validate required fields for each message type. - Save the message to the database with a timestamp. - Respond with a success or error message based on the save operation. Database Tasks - Define Message Schema for storing chat messages: - Schema Fields: - `courseId`: ID of the selected course. - `senderId`: ID of the message sender. - `recipientId`: ID of the recipient for private messages. - `message`: Text of the message. - `senderType`: Type of the sender (student or instructor) - `teamId`: ID of the team (for team-specific messages). - `timestamp`: Date and time when the message was sent. - Create database functions for chat messaging: - getChatMessages(courseId): Fetches and returns all messages for a specific course. - getChatRecipients(courseId): Returns all recipients (instructor, students) for the specified course. - saveChatMessage(courseId, senderId, recipientId, message, senderType, teamId): Saves a new message to the database. Front-End Tasks - Build Chat Component: - Components: - Chat Sidebar: Displays course selector, team selector, and recipient list. - Messages Container: Shows the list of messages with sender and timestamp. - Message Input Form: Input field and send button for composing messages. - Add Styling for Chat Components in `Chat.css`: - Style chat container, sidebar, message bubbles, and message input form for a modern chat layout. - Differentiate message types visually (public, team, private). - Implement Course and Chat Type Selection - Add dropdowns and selectors for switching between courses, public/team/private chats. - Update message list based on selected course and chat type. - Implement Message Sending, fetching in message container - Send a new message through the API and update the messages list on success. - Fetch messages for the selected course and chat type when loaded or updated.	2024/11/08	HIGH	MEDIUM	Robert	3

#65	Export Assessment Summary as a Spreadsheet	"As an Instructor I want to export the assessment summary of a selected course to an Excel file, including detailed student data and a visual chart of average scores so that I can easily share, analyze, and archive the assessment results for record-keeping and further analysis"	2	Front-End Tasks - Add Export Excel sheet Button to Assessment Summary Page: - UI Elements: - Export to Excel Button: Positioned on the existing "AssessmentSummary" page - Configure Data Structure for Exporting: -Prepare student assessment data (Student ID, First Name, Last Name, Team, Cooperation, Conceptual, Practical, Work Ethic, Overall, Responses.) - Embed Chart in Excel File: -Capture the Chart.js chart as an image. -Embed the chart image on a new worksheet titled "Chart" in the Excel file. Export Tasks - Implement ExcelJS Workbook and Worksheet Creation: - Initialize a workbook with two sheets: - "Assessment Data" for the student data. - "Chart" for the captured assessment chart. - Export and Save Excel File Using File-Saver: - Generate an Excel file buffer and save it as ${selectedCourseData.courseCode}_Assessment_Summary.xlsx.	2024/11/09	HIGH	MEDIUM	Yousef	3