Mark as done

Read

Please read the following instructions carefully. If any part is unclear or you have questions, contact the designated teachers through Microsoft Teams direct message.

If you are struggling to understand the Course Assignment instructions, contacting your teachers for help is perfectly acceptable.

Please understand that the teachers **cannot give students the answers or step-by-step instructions on how to implement course assignment requirements,** as the Course Assignments, Projects and Exams are graded assessments. Teachers can, however, **give clarification on Course Assignment instructions** if needed.

The course assignment will be **graded on a "pass / not pass" basis.**

Repository

Only commits made in the GitHub PR will be considered for grading. Ensure all your work is committed **before the course assignment deadline. After the deadline,** you cannot commit any more code to your PR. Any attempts to commit after the deadline will be automatically rejected.

Please ensure you give the teacher access to your GitHub repository if it is not public (Github username: cnnrbrn).

Submission

1. All commits must be pushed to your GitHub repository **BEFORE** the course assignment deadline.
2. The link to your **PR** must be submitted on Moodle.

Commits or submissions past the deadline will not be considered for grading.
Late submissions will not be accepted, and there will be no exceptions to this rule.

Failure to submit the PR link on Moodle OR push your code to the PR will result in a not passed grade

Development Platforms Course Assignment

Goal

Develop a functional news platform that demonstrates your understanding of modern web development practices, user authentication, and data management through either backend API development or full-stack implementation with modern tools.

Learning Outcomes

- Has knowledge of services and APIs used to deliver integrated full-stack solutions
- Has knowledge of processes and tools that are used in front-end projects with cloud-based backend platforms and self-hosted APIs
- Is familiar with organizational requirement and workflows in web development powered by modern, simplified database solutions
- Has insight into his/her own opportunities for development through interdisciplinary collaboration using platform-as-a-service tools and APIs
- can plan and carry out tasks linked to configuring cloud-based and self-hosted APIs
- can plan and carry out programming that accesses and updates data hosted in databases provided by platform-as-a-service software
- can plan and carry out full-stack solutions by leveraging modern backend services
-

Brief

Build a news platform where users can browse and submit news articles.

You can choose between two implementation paths:

- **Option 1:** An Express.js API backend with authentication and database integration
- **Option 2:** A frontend application using Supabase for the backend services

Project Requirements

Core Functionality (Both Options)

Your application must include:

Public Access:

- Anyone can view the list of news articles
- Articles display title, body, category, and submission date

User Authentication:

- User registration with email and password
- User login

Article Management:

- Only authenticated users can submit news articles
- Article details: title, body, category (submission date can be automatic)
- Articles automatically tagged with submitter (logged-in user) information

Implementation Options

Choose one of the following paths:

Option 1: Express.js API

Build a REST API with Express with no frontend required.

Technology Stack:

- Express.js with TypeScript
- MySQL database with mysql2
- JWT authentication with bcrypt password hashing
- Basic validation
- Simple error handling

Key Implementation Requirements:

- Use Express Router to organise endpoints
- Implement JWT authentication middleware
- The create article route must be protected by authentication
- Include proper error handling and status codes
- Use parameterised queries for SQL injection prevention

Required Endpoints:

- **POST /auth/register** - User registration
- **POST /auth/login** - User login (returns JWT)
- **GET /articles** - View all articles (public access)
- **POST /articles** - Submit new article (protected, requires JWT)

Database Tables:

- **users** (id, email, password_hash, created_at)
- **articles** (id, title, body, category, submitted_by, created_at)

Option 2: Frontend with Supabase

Build a frontend application using Supabase for the backend services.

Technology Stack:

- Supabase
- Supabase JavaScript client library
- HTML, CSS, and JavaScript. You can use libraries like Tailwind and React, or do everything with vanilla CSS and JS.

Key Implementation Requirements:

- Implement Supabase authentication with email confirmation
- Create a responsive frontend with proper navigation
- Use Supabase database for article storage

- Implement Row Level Security (RLS) policies
- Show/hide UI elements based on authentication state:
  - The login and register links should not be visible when the user is logged in
  - The create article link should only be visible when the user is logged in

Required Features:

- User registration and login forms
- Article browsing interface
- Article submission form (authenticated users only)
- Proper error handling and user feedback
- Responsive design

README

Your README should include:

- Installation and configuration information and instructions on how to run the project.
- A **Motivation** section that explains:
  - why you chose the option you did. If you chose option 1, you could explain that you enjoyed developing the server-side code, for example. If you chose option 2 you could explain you enjoyed developing a full-stack site.
  - what you liked about the development process and what you didn't enjoy.
  - what you found difficult.
  - what you think the benefits of developing a custom API are versus using a SaaS option like Supabase, or vice versa.

The Motivation section does not need to be long.

Deliverables

- Create a new repo in your GitHub account called **development-platforms-ca**.
- You can work on branches, but the final submission should be on the **main/master** branch.
- If doing option 1, the repo will include:
  - The Express project
  - The exported database in a **.sql** file
- If doing option 2, the repo will include:
  - the frontend code for the project including working Supabase configuration and integration.
- Both options must include the Motivation section in the README.

Grading Criteria

Option 1

- the GET /articles endpoint returns a list of articles
- the POST /articles endpoint is protected by auth and creates an article
- the POST/ auth/register endpoint registers a new user
- the POST /auth/login endpoint logs a user in and returns a token
- the exported database SQL file contains the correct, working schema for the application
- the API code is sensibly arranged with ES modules and Express Router
- the endpoints contain validation and basic error handling

Option 2

- a frontend project in either vanilla JS or React
- the project must be completely responsive
- the home page will return a list of articles
- there is a page with a form for creating articles
- the create articles page must be auth-protected
- there is a login page with a form that logs a user in
- there is a register page with a form that registers a user
- any error messages should be displayed in the UI to the user

Resources

Here is a video on how to export a database using MySQL Workbench.

Jump to...

◀ Development Platfo...

Development Platforms - C... ›

Contact site support

You are logged in as Sergiu Dumitru Sarbu (Log out)

Data retention summary

Policies

Get the mobile app

Powered by Moodle

Follow Us

Privacy Policy