Web Programming SODV2201 Assignment and Project Work 2025

Assignments/Project overview

Bow Course Registration Fullstack Web Application

This project involves building a **full-stack web application** that will allow students in the Software Development (SD) department to register for courses online. You'll work in groups of 3 or 4, and your project will be completed in three phases: **Assignment 1 (Frontend), Assignment 2 (Backend), and Final Project (Full Integration)**. Each phase builds upon the previous one.

Group Project Information

- **Group Size**: Work in groups of 3 or 4.
- **Joining a Group**: Go to D2L's communication group tab to join a group. If you don't join before Assignment 1 is due, you will receive a **zero**.

General Overview of the System

The Bow Course Registration system is designed for the **SD department**, offering different programs and courses. The main goal is to create a web application where students can:

- View available programs and courses.
- Register for courses based on their selected program and term.
- Admins can manage courses and view student details.

SD Department Programs:

- Diploma (2 years)
- Post-Diploma (1 year)
- Certificate (6 months)

Terms:

- **Spring**: March - June
- **Summer**: June - August
- **Fall**: September - December
- **Winter**: January – March

Each program and course will display details such as:

- **Programs**: Program code, department, term, start/end dates, fees, and descriptions.
- **Courses**: Course code, name, term, start/end dates, and descriptions.

Non-User Features:

1. Non-users can view all programs and courses.
2. Non-users can **sign up** for the system to become students.

- **Signup Page Details:**
  When signing up, students must provide:
  1. **First Name**, **Last Name**, **Email**, **Phone**, **Birthday**, **Department** (only SD department will be available), **Program**, **Username**, and **Password**.
  2. After signing up, the system generates a **Student ID** and redirects the student to either a **login page** or a **welcome page** (you can choose).

Student Features:

1. **Dashboard**: Displays information such as first name, student/admin status, student ID, department, and program.
2. **View Profile**: Students can view their profile information.
3. **Term Selection**: Before registering for courses, students must choose a term (Spring, Summer, Fall, or Winter).
4. **Course Registration**: Students can register for **2-5 courses** per term and can't register for the same course twice in the same term.
5. **Add/Remove Courses**: Students can add or remove courses from their selection.
6. **Search Courses**: Search for courses by name or course code.
7. **Submit a Contact Form**: Students can send messages to the admin.

Administrator Features:

1. **Dashboard**: Displays information such as first name, student/admin status.
2. **View Profile**: Admins can view their profile information.
3. **Create Courses**: Admins can create new courses by providing details such as course name, start/end dates, and more.
4. **Edit Courses**: Admins can edit courses on the system.
5. **Delete Courses**: Admins can remove courses from the system.
6. **Search Courses**: Admins can search for courses by name or code.
7. **View Registered Students**: Admins can see a list of students registered for each program.
8. **View Submitted Forms**: Admins can read to messages sent by students.

Sample Program Data:

- **Software Development - Diploma (2 years)**
  - Term: Winter
  - Description: A comprehensive two-year software development diploma program designed to equip students...
  - Start Date: September 5, 2024
  - End Date: June 15, 2026
  - Fees: $9,254 domestic / $27,735 international

- **Software Development - Post-Diploma (1 year)**
  - Term: Winter
  - Description: Jumpstart your tech career with our one-year post-diploma program in software development....
  - Start Date: September 5, 2024
  - End Date: June 15, 2025
  - Fees: $7,895 domestic / $23,675 international

Direction

The objective of this use case is to give you an idea what aspects to include in your project. As a starting point you can use what is explained above and add different features and modification to it to meet your front-end and backend application development need as you learn progressively.

This project has 3 parts Assignment 1 and Assignment 2 and a final project. Assignment1 outcome will be used as input for your Assignment2. Then the combined final submission of this two- assignment work will be taken as your course project work. There will be a set of special features in each phase of this assessment.

Project Parts Breakdown:

Assignment 1: Frontend Development

- **Goal**: Design and create the frontend using React.js. Store all data (students and courses) in arrays or objects.

- **Steps**:
  1. Understand Web Architecture: Learn the basic structure of how web apps are designed.
  2. Wireframes: Create sketches of your web pages to guide your design.
  3. Implement in React.js: Use React.js to build the frontend, paying attention to naming conventions and file structure.

- **Key Design Considerations**:
  - What pages do I need (e.g., signup, course listing)?
  - Which parts should be reusable components?
  - How should I handle state and state management?
  - How will the frontend connect to the backend APIs in the next phase?

- **What to submit for A1**:

  You are expected to make progress on your site throughout the semester. You must meet the following conditions as part of A1 submission:

  1. Project file zipped and uploaded on D2L A1 dropbox (10-to-15-minute demonstration video recording)
  2. Must be capable of running on a local machine.
  3. It should have full functionality of react based UI and user interaction feature.
  4. Each student must demonstrate the submission if requested.

Assignment 2: Backend Development

- **Goal**: Build the backend using Node.js and a database (SQL/MySQL or MongoDB).

- **Steps**:
    1. Design the Database: Create a database schema and populate it with sample data.
    2. Set Up the Server: Use Node.js to build a web server.
    3. Create APIs: These APIs will allow the frontend to retrieve and update data in the database.
    4. Implement Security: Include login functionality and session management.

- **Assignment expectation**
    - Prepare the backend design of your website. Make sure it aligns to what you built on A1.
    - Implement backend web server using Node.Js and SQL/MySQL or MongoDB server, .JSON file etc as data storage.
    - Create different local APIs to process data from your database.
    - Discuss your ERD diagram or file structure to make sure the schema accommodates the features of your front-end website. You should be able to run, connect to, and query your dataset/database and populate the database with sample record/update the dataset.
    - Which API should you implement to retrieve information to/from the Bow course registration system?

- **Assignment2 Requirements**

  In this part of the assignment2, you should work on:
    1. Working database schema implementation populated with sample data or structured dataset source.
    2. Working web server setup
    3. Backend web application and database/dataset integration
    4. Local APIs to fetch data to/from your database/dataset (you can use **postman** for testing)
    5. Security and validity feature
    6. Login or session management

- **What to submit for A2**

Consider the following to submit as A2 file:

1. Node.js project file zipped and uploaded on D2L A2 dropbox (10-to-15-minute demonstration video recording)
2. Must be capable of running on a local machine.
3. It should have full functionality of fetching data from your data storage, validation, and certain level of security implementation by the time this assignment is due (eg. **bcrypt** for password encryption).
4. Each student must demonstrate the submission if requested.

Final Project: Fullstack Integration

- **Goal**: Integrate the frontend from Assignment 1 with the backend from Assignment 2.
- **Key Considerations**:
    - Ensure the APIs are working properly with the React UI.
    - The system must allow students to register for courses, and admins to manage tasks like adding or removing courses.
- **Implementation**:

In this final project phase, you are expected to integrate your front-end work from A1 with your A2 backend project. Then instead of using postman or backend html page to utilize your backend data storage and API you will use your react UI.

Much like your frontend, you should think carefully about your design before you start implementing any code. Consult your sketch and consider the following questions:

1. Which API should get information to/from Bow registration system?
2. Proper use and implementation of APIs to work on Bow registration system UI.
3. What is the input and output of any functions I need? Are there edge cases?
    1. How should I structure my end points?
    2. How will I manage login and sessions?
4. Is my design modular?
    1. Does this design exhibit low coupling and high cohesion?

- **What to submit for Final project**

To pass A2, your back end must meet the following conditions:

1. Fullstack project that contains both front-end react project and backend node.js project file zipped and uploaded on D2L project dropbox
2. Upload a 10-to-15-minute demonstration video recording
3. Must be capable of running on a local machine.
4. Full functionality to allow students to search and register for a course. Admin to be able to work some admin related task. Be able to add or remove course, be able to get message from students etc.
5. Each student must demonstrate the submission if requested

Marking rubric

Project will be marked according to the following rubric:

	4 10 **Exceptional**	**3 7** **Good**	**2 5** **Fair**	**1 3** **Poor**
Design Quality **5**	All major functionality adheres to design best practices for the front-end framework	Most functionality is well designed, but there is one major oversight or a few small oversights. 7	Design has several major flaws, but still uses some aspects of the framework best practices. 5	Design did not use framework or used it trivially.
API Calls **10**	Front end uses all mock server endpoints correctly.	Front end uses most mock server endpoints correctly. 7	Front end uses at least one mock server endpoint correctly. 5	Front end does not use mock server.
Implementation Quality **10**	Implementation uses chosen library effectively, handles errors, and is reasonably efficient.	Implementation uses library, but not as effectively as possible **OR** does not handle errors reasonably **OR** is grossly inefficient. 7	Implementation barely uses library, or uses it incorrectly. 5	Library is not used, or is trivially used.
Style and Maintainability **10**	Code follows industry standard conventions for organization, documentation, and style. Deviations from industry standard are trivially small.	The code does not follow industry standards, but is still easy to follow. 7	Code can followed, but it is difficult due to unconventional style, organization, or documentation.5	Code is almost impossible to follow due to poor style, documentation, or organization.