Software Engineer Assessment

There will be two (2) tasks required for you to demonstrate. Submission must be done on or before 9th of June, 2025.

Submission Link : **https://auesurvey.questionpro.com/t/AOqWxZ6Qhj**

Task 1: Faculty-Student Assignment & Assessment Portal

Objective

- Design and implement a full-stack web application using .NET Core MVC that supports:
- Faculty and Student role-based logins
- Assignment creation and submission
- Grading and assessment creation
- Data visualization of performance metrics

Key Skills to Evaluate

- Database Design (MS SQL Server with stored procedures only)
- RESTful API Design (for AJAX/front-end or service consumption)
- N-Tier Project Architecture
- ORM with Dapper
- Frontend Frameworks (Bootstrap, jQuery, or similar)
- Data Visualization (Power BI embedded, Chart.js, or similar)
- CI/CD (GitHub Actions or Azure DevOps pipelines)
- Unit Testing
- (Bonus) Integration with any AI tool/service (e.g., OpenAI for auto-assessment feedback)

Functional Requirements

Faculty Interface

1) Secure login and role-based access
2) Dashboard with:
    a) Student listing (retrieve from DB)
    b) Create Assignments
        i) Title, Description, Due Date/Time, File Upload
    c) Create Assessments
        i) Linked to Assignments
        ii) Multiple grading criteria (e.g., Clarity, Completeness)
        iii) Score range per criterion (e.g., 1-10)
        iv) Remarks entry
    d) View performance charts of all students (line chart, pie chart)

Student Interface

1) Secure login and role-based access
2) Dashboard with:
    a) View Assignments

    i) Submit before deadline (upload file or link)
  b) View Graded Assessments
    i) See criteria, score, remarks
  c) Visualized performance (individual)

Technical Requirements

1) **Backend:**
  a) .NET Core (v6 or above)
  b) MVC Architecture
  c) Use **Dapper** for data access (via stored procedures only)
  d) SQL Server for database
  e) Authentication via Identity or custom role management
2) **Frontend:**
  a) Use **Bootstrap** or any UI Framework
  b) AJAX calls for dynamic content (jQuery/Fetch)
  c) Chart.js / Power BI / Any visualization tool
3) **CI/CD:**
  a) Code must be hosted on **GitHub** or any online repository
  b) Include:
    i) Build pipeline (unit tests required)
    ii) Release pipeline (optional: deploy to Azure Web App or IIS)
4) **Unit Testing:**
  a) Write tests for service and data layers
5) **Bonus:**
  a) Optional integration with AI tool
    i) Example: Use OpenAI API to auto-generate feedback on assignments

Task 2: University Course Scheduling and Section Optimization System

Objective

- Design a scheduling engine that generates an optimized timetable for university course sections (lectures and labs), adhering to constraints.

Business Requirements

1) **University Operating Days:** Sunday to Thursday
2) **Operational Hours:** 8:00 AM – 4:00 PM
3) **Section Offering Logic:**
   a) Number of sections = Based on eligible students and room/lab capacity
   b) Some courses include both **lecture** and **lab** components
   c) Lab capacity must be respected independently of lecture capacity
4) **Lecture Scheduling Rules:**
   a) 3 lecture hours must follow either:
      i) **Scheme A:** 1.5 + 1.5 hours (2 non-consecutive days)
      ii) **Scheme B:** 1 + 1 + 1 hours (3 non-consecutive days)
   b) Section offerings must be fairly distributed between the two schemes *(e.g., 6 sections = 3 in A, 3 in B)*
5) **Lab Scheduling Rules:**
   a) Labs cannot be split across days (e.g., 2-hour lab must be continuous)
   b) Lab must not conflict with its respective lecture timings
   c) If only one lab section exists for a course, it must not overlap with any of its lectures

Expected Output

1) Optimized, non-overlapping **schedule for each course section**, including:
   a) Assigned room/lab
   b) Day(s) and time slot(s)
2) **Export to Excel** (grouped by course, section, room/lab)
3) **Visualization Dashboard**:
   a) Heat map showing room usage
   b) Bar or Gantt chart showing time allocation for each course

Technical Requirements

1) **Backend**:
   a) .NET Core (v6+), MVC Architecture
   b) Dapper for DB access (Stored Procedures only)
   c) Scheduling logic can be implemented using algorithmic approach (greedy/backtracking) or optimization libraries
2) **Frontend**:
   a) Bootstrap (or any lightweight UI)
   b) Visualization using Chart.js, Power BI Embedded, or Syncfusion
3) **Export**:
   a) Downloadable Excel file
4) **CI/CD**:
   a) Hosted on GitHub or any online repository
   b) Build pipeline with tests

c) Release pipeline (optional Azure deployment)

5) **Unit Testing**:
   a) Validate scheduling logic with edge cases
6) **Bonus**:
   a) Schedule optimization suggestion using AI/ML (optional)

Sample Data :
Feel free to add more

Course Code	Course Name	Lecture Hour	Lab Hour	Eligible Students
CT 100	Intro to IT	2	2	90
CT 105	Intro to AI	2	2	80
EN 100	English 1	3	0	106
EN 200	English Communication	3	0	90
PH 100	Physics 1	3	1	85
PH 200	Physics 2	3	0	70

Classroom	Type	Capacity
CR 100	Lecture	35
LAB 101	Lab	15
CR 102	Lecture	35
LAB 103	Lab	15
CR 104	Lecture	30
LAB 105	Lab	40