LEGISLATIVE ASSEMBLY
of BRITISH COLUMBIA

Written Assessment Instructions

Competition LA252618 | Full Stack Developer

Thank you for your interest in employment with the Legislative Assembly of British Columbia. Your application has been shortlisted for the Full Stack Developer position, Competition LA252618.

Thank you for taking the time to complete this work. This assignment is designed to assess your technical skills and experience in a realistic, practical way with reference to tasks and challenges you may face in the role. We estimate that the assignment should take no more than two hours of your time but you have until the deadline below to complete it.

You may use whatever tools and resources you consider appropriate and whatever languages, frameworks, databases, and other technologies you know best. You can work locally, and we do not expect running code. Where relevant, please include brief references in your comments or commit messages indicating when code was borrowed or adapted from external sources, whether from your own libraries, open-source examples, AI assistance, or other resources.

Please submit your completed assignment as a ZIP file to Human.Resources@leg.bc.ca. You must return your assignment no later than 9:00AM on Monday, June 9th . The document name and email subject line should be: LastName, FirstName - LA252618 Full Stack Developer Written Assignment. Your submission will be time-stamped through your return email.

Assignment Breakdown

Out of a possible 100 points the assignment is in three parts:

- Part 1: Database and API Design (40 points)
- Part 2: Coding Task (40 points)
- Part 3: Short Questions (20 points)

Part 1: Database and API Design (40 Points)

Entity Relationship Diagram

Produce a simple entity relationship diagram (ERD) which describes a system that supports:

- Posts
- Comments
- Hashtags

Please include the references or relationships between entities in your diagram. This can be hand drawn or generated in software. Please submit this as a PDF or an image.

Database Creation Commands

In the language of your choice write code which would create the database you have described above. No need to include every detail, just accurate table/collection/relationship/structure definitions as appropriate. Please provide that code in a plain text file.

Query

Given your database structure, write a set of queries which retrieve:

- A single post
- All comments on that post
- All hashtags associated with the post

Using comments or pseudo code, describe how you would nest the comments on the post. Please provide the queries, nesting code and comments in a plain text file.

HTTP Flow

Write out in plain text or pseudo-format an HTTP exchange which creates a new comment on an existing comment to the original post including:

- Request(s), method, URL, headers, and sample request bodies
- Expected response(s) including sample response bodies.

You do not need to implement it, just document the expected HTTP request and response clearly. Please provide this in a plain text file.

Part 2: Coding Task (40 Points)

Using any language or MVC framework, write pseudo code to describe an API endpoint that would perform the following tasks. Please include comments or documentation explaining your logic and reasoning.

1. Receives a multipart form-data POST to /api/posts that includes
   - Title
   - Body
   - Hashtags
   - A file attachment using the field name ‘file’.
2. Securely processes the attachment
   - Confirms that the file is one of (PDF, JPG, PNG)
   - If the file check fails, return the appropriate HTTP response code with a JSON response body
   - If the file check passes, review it for malware in a commented code stub.
3. On success
   - Return the appropriate HTTP response.

Version Control & Continuous Integration/Continuous Delivery:

In a plain text file, write out the appropriate Git commands and messages as though you were committing the above code and other files to a shared repository.

Include a basic CI configuration that:

- Runs tests (if any)
- Lints code or checks formatting

Part 3: Short Questions (20 Points)

For each question, write a few sentences or bullet points which respond to the question. Provide your response in a text file or PDF.

Time Math

As you know British Columbia includes regions with different time zones. What strategies do you employ in your code to accurately manage time differences, particularly in relation to scheduled and live events?

Logging

Logging automated processes is important for debugging, maintenance, and auditing. What should and should not be included in log files? What factors should be considered when implementing logging for internal systems versus those accessed by the public?

Technology Selection and Recommendation

What points would you consider and what approach would you take when selecting and recommending a technology or framework be implemented in an environment like the Legislative Assembly?

Technical Debt

In the context of full stack development, define technical debt. Provide examples of technical debt you have incurred and how it was mitigated or delt with.

Submission Guidelines

Please submit the following materials in a well-organized and clearly named ZIP file:

- ERD PDF or image
- Database creation script
- Query
- HTTP flow
- Endpoint code
- Git commands and messages
- Continuous Integration configuration
- Short questions