Code Challenge Notice, Instructions & Rules

Government Contact: steve.chapman@gov.bc.ca

This notice is dated November 15th, 2023 at 9 a.m. (the "Notice Date").

Congratulations - you are a Shortlisted Candidate eligible to participate in the Code Challenge.

Rules and Instructions

Please be advised of the following rules and instructions:

1. These code challenge rules and instructions apply only to Shortlisted Proponents and are part of the Request for Proposal (RFP).
2. Shortlisted Proponents will have at least three (3) Days from the Notice Date to complete the code challenge. The deadline to complete the code challenge in accordance with these rules is **9:00 a.m. Pacific Time on Saturday, November 18th, 2023** (the "Deadline").
3. The Shortlisted Proponent's code challenge submission Deliverable (defined below) must be received by the province (as provided for by these instructions) and be deposited and located in the applicable Repository before the Deadline, failing which such submission will not be eligible for evaluation and the associated Shortlisted Proponent Proposal will receive no further consideration and such Shortlisted Proponent will be eliminated from the RFP competition.
4. Only the Proponent Resources that were put forward in a Shortlisted Proponent's RFP Proposal are eligible to participate in the Code Challenge.
5. The Shortlisted Proponent Resources will be sent invites via GitHub to join this private repository.
6. As of the Notice Date, the code challenge issue has been created in this private repository, under the "BCDevExchange-CodeChallenge" organization.
7. Shortlisted Proponents may direct clarifying questions by creating an issue in the applicable GitHub Repository. Any such questions must be received by the Government Contact **before 4:00 p.m. Pacific time on Thursday, November 16th** (the "Code Challenge Questions Deadline").
8. The Province reserves the right to amend this notice before or after the Closing Time, including changing the Deadline or the Code Challenge Questions Deadline upon notice to all Shortlisted Candidates.

9. The Shortlisted Proponent must complete all the following tasks and the Deliverable and as such they must be deposited and received in the applicable Repository by the Province in the form specified by this notice before the Deadline:
10. Complete all code changes required to complete the code challenge (the "Deliverable"); and
11. Attach an Apache License 2.0 to Deliverable.
12. The rules and instructions set forth in this notice are in addition to any rules, terms and conditions set forth elsewhere in the RFP.

Problem Statement:

Emergency Management Scenario:

BC Gov Employees want to enroll for possible deployment as volunteers in case of disasters such as a wildfire, flood, earthquakes, etc. Employees will need to submit an application describing what they have to offer and any limitations on their ability to volunteer. After submitting an application, they get confirmation that they have been registered as a qualified volunteer.

BC Gov maintains a database of types of disasters that require a deployment of volunteers to add surge support. The database of disasters describes skills that are needed for each category of severity.

When a disaster occurs, a web page is available that lists volunteers who are qualified and able to provide surge support for that disaster [the web page can be public].

We would like a web interface that can collect the volunteer data and store it and provide a page for each disaster that includes a list of suitable volunteers.

User flow or process:

- The application should allow users to fill out a form.
- The front-end should interact with an API to read and write data from and to the database.
- The front-end needs to provide a result page after a form is submitted and display data from the database.

Features:

- Automated deployment using GitPod (https://www.gitpod.io/docs/introduction), OpenShift or similar technology
- Form submission page (Creative designs are welcomed)
- Persistent storage of submission data
- Display error messages to the user
    - Display a message if front end error occurs (must do at least one front end validation)
    - Display a message if an API level error occurs (must do a least one API validation)
- View submission page (read only)
- Testing framework to automate regression testing

Technical Requirements:

- API follows a best practice protocol for managing the data
- Use of modern languages is recommended
- Error handling on the submit call
- Rollback transactions if errors exist in the submission
- Test automation framework demonstrates checks for front end and API code quality and code correctness

Out of Scope:

- State management (refresh of the browser will clear values and start over)
- Identity management (login, OIDC/OAuth) capabilities
- Cloud hosting
- Messaging Queues
- Role based access controls
- Data exports
- External API calls

Deliverables:

- Include justification for the selected technologies and libraries with diagrams showing how they interact
- Build/deploy environment configuration (e.g., GitPod, OpenShift or similar technology)
- A README file explaining how to set up and run the application.
- Service design artefacts (research data can be fictional) that give clarity on the target users and benefactors.
- UX artefacts to guide the flow and the look and feel of the application

- A plain language end-user support artefact in the form of a one-page FAQ article suitable for publication via a support wiki that provides end-user self-service support for (fictional) requests from end-users who find they cannot map their skills into the provided interface: describe the issue and propose an end-user workaround. Additionally and separately, compose a brief JIRA-ready user story for a feature request that could be passed to the product team for development of a feature enhancement that would resolve this issue.
- Demonstration media (can be a screen recording, pdf, PowerPoint or other media that walks through the solution). Please name the file “demo.mp4” or “demo.pdf”
- In the demo media, describe any test automation tools that were used, identify the level and type of testing implemented and what software development processes were employed to ensure testing was successfully delivered alongside and in support of the functional requirements.

Evaluation Criteria:

- (30 points) Completes requested application features
- (20 points) Correct implementation of the technical requirements.
- (10 points) Adherence to best practices for code quality and security
- (15 points) Usability and appearance of the application
- (15 points) Clarity of the deliverable documentation and artefacts
- (10 points) Overall quality of the application and alignment of deliverables