|  |
| --- |
| **BC Public Service, Citizens’ Services, Exchange Lab - Digital Office**  **Req#92534 – ISL 27R – Full Stack Developer**  **Written Assignment Phase**  **Your Name: Yiguang(Mike) Dong**  **Date: Dec 2, 2022**  Your completed written exercise must be returned via e-mail to [DIGITAL.TALENT@GOV.BC.CA](mailto:DIGITAL.TALENT@GOV.BC.CA) **no later than 16:00 (Pacific Time) on Saturday, December 3rd** 2022. The written exercise administrator will confirm the receipt of your written exercise. Any responses after received after **16:00 (Pacific Time) on Saturday, December 3rd**, 2022 will not be considered and your application will be withdrawn from the competition.  **Purpose:**  The purpose of this written exercise is to test your skills and knowledge along with your analytical and judgement abilities. Your written communication skills will also be evaluated.  **Assumptions:**  While you answer questions, it may be necessary to make some assumptions. If any assumptions are made, please note them along with the decision criteria that led to the assumption.  **Responses to Questions:**  Please keep your responses concise (i.e., no more than one or two pages for each question). You may use the internet or other sources to do any research for your responses. However, your responses must be in your own words. If you quote any material from any external sources (e.g., websites, books, etc.) please reference each source in your responses.  Responses should be submitted in either Word or PDF. For modifications to the code provided, please save it under a new name and attach it to your response.  **Good Luck!** |
| **Question 1:**  The Ministry of Transportation and Infrastructure’s (MoTI) Information Management Branch (IMB) and the DriveBC program area have developed an open API for public access to road condition and event data in B.C. The Open511 API complies with the Open511 specification and is licensed under the Open Government License in B.C. Please refer to the following link for more information about the DriveBC Open 511 API:   * DataBC Catalogue:   <https://catalogue.data.gov.bc.ca/dataset/open511-drivebc-api> * Open511 Help:   <http://api.open511.gov.bc.ca/help>   Using the language and libraries of your choice, build a one-page web app that accesses the DriveBC Open511 API to satisfy following user stories. Please build a local development environment that can be deployed on another machine with a one or two-line command. This web application includes a backend, a database, and a frontend web application.  User stories:  #1.  As a user, I want to navigate a web-based application, so that I can find events of interest.  #2.  As a user, I want to save the data retrieved from the REST APIs to a database for the further process.  #3.  As a user, I want to filter out events by Area, Severity, Event Type and Start Date  #4.  As a user, I want to provide two REST API endpoints to give data access to other applications. The REST API endpoints include creating new events and getting highest severity events by Areas.  #5.  As a user, I want to monitor API usage data based on the API calls from other applications. |
| **The website `http://open511.org/documentation.html` doesn't show documents, but a domain name alert page. I did google `Open511` and found `https://github.com/open511`, which gives me the documents about Open511 API.**  **#1. The APIs can be called across domains, so we can send requests to `api.open511.gov.bc.ca` from the browser.**  **#2. Because we send requests to `api.open511.gov.bc.ca` from the browser, the data we want to save needs to be sent from the browser too. We can create an API for saving data.**  **#3. The API has no parameter for `Start Date` filter, I use `Create Time` instead and return the data after it.**  **#4. The `creating new events` API is the same feature as above, But these two APIs are for other applications, so I make them separate.**  **#5. The calling from other applications will be saved in the database for monitor reasons in the future, including time, client, and API.**  **Technologies used: Vue.JS, Element-Plus, jQuery, Google Map, LeafLet, Python, MongoDB, Nginx, Docker, etc.**  **Please access <https://github.com/dongyg/req92534> to download the code and follow the README. Once all things go well, please open a browser to access <http://localhost:8027>. The web page should be like below. Thank you!** |
| **TOTAL (/10):** |
| **Question 2:**  We had some REST API endpoints built as below for a very restricted bandwidth environment, and we are going to upgrade these endpoints to support new requirements in the upcoming version.  Getting all vehicles in the province  GET [http://drivebc.ca/api/vehicles](http://drivebc.ca/api/vehicles/search/%7bid%7d)  Checking if a vehicle exists:  GET [http://drivebc.ca/api/vehicle/search/{id}](http://drivebc.ca/api/vehicle/search/%7bid%7d)  Creating a new vehicle  POST <http://drivebc.ca/api/vehicle/create>  Getting a vehicle with vehicle id 123  GET <http://drivebc.ca/api/vehicle/123>   1. Identify the issues in the API design and 2. Describe how you would fix the issues and make improvements. |
| **Checking if a vehicle exists:**  **GET http://drivebc.ca/api/vehicle/search/{id}**  **No need to name a `search` API for checking, return the HTTP 404 instead on the API `/api/vehicle/{id}`.**  **Creating a new vehicle**  **POST http://drivebc.ca/api/vehicle/create**  **No need to name a `create` API for creating an item, do create on `/api/vehicles` API.**  **Normally, we design RESTful APIs for CRUD like this:**  **GET /vehicles – list vehicles**  **POST /vehicles – create a vehicle**  **GET /vehicle/{id} – get a vehicle**  **PUT /vehicle/{id} – update a vehicle**  **DELETE /vehicle/{id} – delete a vehicle**  **PUT /vehicle/{id}/operation - other update operations**  **If we are upgrading the existing APIs which is for a very restricted bandwidth environment, we should consider some things**  **1. Reduce unnecessary data transit, such as the path of the APIs, we can make them shorter, eg: `/v/{id}`**  **2. Compatibility, we should ensure APIs compatibility when upgrading the existing APIs** |
| **TOTAL (/10):** |
| **Question 3:**  We are running into some performance issue when integrating third party services within our own API requests. The user must wait for the response from our APIs, and as such, forcing the user to have to wait for a long time.  How would you go about avoiding this? Name any relevant technologies if appropriate |
| **It depends on what we use the third party services to do, and how we do it.**  **1. If the processing can be asynchronous, that should be performed**  **2. If the data from the third party services is not real-time data, we can cache the data properly**  **3. If third party services can be requested on the client side, with no safety problems, etc, do it**  **4. If we can't make the process faster, do some work on UI and make the users feel not so slow** |
| **TOTAL (/10):** |
| **Question 4:**  How do you measure good code and bad code? Please describe the ways for improving your code quality to mitigate risks and how do you conduct the best practices in your experience. |
| **1. Good code should be well-structured, easy for understanding**  **2. A good design is the premise for well-structured coding**  **3. Code comments are necessary, and very useful sometimes**  **4. Code shouldn't be too long for a single function**  **5. Variants and functions should be named properly**  **6. Write unit test**  **In my experience, guidelines will be very helpful, making rules to show everyone what should be followed. Plus, code review is necessary, find problems in time to correct them in time.** |
| **TOTAL (/10):** |
| **Question 5:**  When designing and implementing a web application with database and scale capability, what are some of the programming techniques you would use for optimizing performance to achieve the goal? |
| **Programming**  **1. Reduce the database query times, such as avoiding the database query in a loop**  **2. Optimize the database schema and indexes**  **3. Avoid querying all fields from the database, and avoid or reduce loop data processing at the backend**  **4. Use a cache for high-frequency access data, such as Memcache or Redis**  **Infrastructure (cloud)**  **1. Choose the proper host provider and location**  **2. Set the proper bandwidth environment**  **3. Set the proper host configuration**  **4. Use the provider's products properly, such as Amazon RDS, ECS, ElastiCache, etc**  **PS: We use those Amazon products on our project Eugris.com to make the system stable and scaled.**  **Infrastructure (host)**  **1. On a single-host system, set multiple application instances**  **2. Use load balance such as Nginx upstream, and distribute requests to different application instances**  **3. Ops scripts to monitor and manage application instances**  **PS: I use these methods on Vansky.com to make the website stable and scaled, achieving nearly 1 million PV per day on a single host.** |
| **TOTAL (/10):** |
| **TOTAL ASSIGNMENT SCORE (/50):** |