**Software Developer Course Assessment**

**Quantitative Assessment Practice #3 *Integrated***

**Course Name(s): Full-Stack JavaScript**

**Database Programming and Data Processing**

**Current Session: July 2022**

**Introduction:**

The purpose of this assessment is twofold; first, to help us understand how the class is doing in terms of the course material that we have covered during the previous couple of weeks. The **main** purpose of this assessment is for us to improve our approach to review and ensure that what we’re currently doing is an effective teaching strategy for the students. Completion of this assessment is **mandatory - if you don’t submit a solution, it will be marked as incomplete. You must complete a minimum of 75% of your assigned QAPs per course – otherwise you will be marked as incomplete for that course no matter how good your other grades are.** If you do submit a solution, it will be marked as complete, as you will receive full marks no matter what your actual performance was – again this is a participation grade.  
  
**Again, the goal here is to help you** all in the best way that we can, so please do be honest when answering the questions related to how long it took, which resources you used, etc. And please ensure that you do your **own** work – don't just copy off a friend to get it done, earnestly do your best with it. If you can’t get it completely working, give us what you have. While it will be graded, the grade will not count against you, it’s just a way for us to see where everybody is, and to know which concepts, if any, we, as a class, may be struggling with.

**Deadline:** Your instructor will determine the deadline for submission for the assessment of your solutions. Please ensure you answer all the questions outlined in the instructions portion of this document as well in your submission.

**Marking:** In this program core evaluation is marked with one of three possible marks: *Incomplete, Pass, Pass Outstanding.* For QAPs, though, where incomplete marks are more important for our own information as well as for the information of the student, we wanted to increase the resolution of our grading system. Therefore, QAPs are marked on a scale of 1-5. The details of this marking system are summarized in the table below.

|  |  |
| --- | --- |
| **Grade** | **Meaning** |
| 1 | *Incomplete.* Student shows severe lack of understanding of the material – solution is heavily incomplete, non-functional, or completely off base of what the assignment was asking for. |
| 2 | *Partially Complete.* Students show some understanding of the material. Solution may be non-functional or partially functional, but the approach is correct, albeit with some major bugs or missing features. |
| 3 | *Mostly Complete.* Student demonstrates understanding of the major ideas of the assignment. Solution is mostly working, albeit with a few small bugs or significant edge cases which were not considered. Shows a good understanding of the correct approach, and is either nearly a feature-complete solution, or is a feature-complete solution with some bugs. |
| 4 | *Complete (Equivalent to: Pass.)* Student shows complete understanding of assigned work and implemented all necessary features. Any bugs that are present are insignificant (for example aesthetic bugs when testing the functionality of code) and do not impact the core functionality in a significant way. All necessary objectives for the assignment are completed, and the student has delivered something roughly equivalent to the canonical solution in terms of features and approach. |
| 5 | *Complete with Distinction (Equivalent to: Pass Outstanding)* The student demonstrates a clear mastery of the subject matter tested by the QAP. The solution goes above and beyond in some way, makes improvements on the canonical solution, or otherwise demonstrates the student’s mastery of the subject matter in some way. A solution in this category would consider all reasonable edge cases and implement more than the necessary functionality required by the assignment. |

**Instructions:**

You are allowed to complete the assessment problems below in whatever way you can but please answer the following questions/points as part of your submission:

1. How many hours did it take you to complete this assessment? (Please keep try to keep track of how many hours you have spent working on each individual part of this assessment as best you can - an estimation is fine; we just want a rough idea.)
2. What online resources you have used? (My lectures, YouTube, Stack overflow etc.)
3. Did you need to ask any of your friends in solving the problems. (If yes, please mention name of the friend. They must be amongst your class fellows.)
4. Did you need to ask questions to any of your instructors? If so, how many questions did you ask (or how many help sessions did you require)?
5. Rate (subjectively) the difficulty of each question from your own perspective, and whether you feel confident that you can solve a similar but different problem requiring some of the same techniques in the future now that you’ve completed this one.

**Problem Background:**

Understanding the Full-Stack end-to-end and the best practices in their design combined with the ability to build a RESTful API with Node, Express, EJS, and PostgreSQL is a desired technical skill. APIs are everywhere and most software technology companies providing services on the internet have many APIs, and related user interfaces, to get at their data.

The problem for this QAP3 is to build an API and simple user interface against the dvdrentals database. An important part of this QAP is to have your node js scripts access the embedded SQL as database VIEWS to retrieve data from the dvdrentals database. If you aspire to only **complete** this QAP I suggest you keep the working database view to a simple two table JOIN. If you aspire to **complete with distinction**, I suggest you create database views with multiple database tables that JOIN in a 1:M or M:M relationship.

Please work alone on this QAP. Ask for help from classmates, the instructor, the KPM, or other resources. Please create an original work.

**Complete the following tasks:**

1. Choose one of the user stories from the list below and develop a full-stack solution;
2. As a customer of dvdrental I would like to have a web page so I can view the films I have rented in the last 12 months.
3. As a store manager at dvdrental I would like to have a web page so I can view the films that have not been returned and the customer who rented the film.
4. As the owner of dvdrental I would like to have a web page so I can view the top ten rental films by rental dollars earned for a particular store.
5. Identify the dvdrental database tables required to fullfill the data requirements of your chosen story.
6. Create a SQL SELECT statement with the required table JOINS to bring all the required data fields together in one SELECT statement.
7. Create a view using the working SQL statement. Test your view with a SELECT \* FROM [view\_name]
8. Create a QAP3 project folder to disk for storing all the related project files. Be sure to create folders to organize all your files into well-organized modular code.
9. Save all SQL queries to disk file(s) into a folder dedicated to SQL.
10. Using NPM packages to build a simple RESTful API and web application in Node.js using the EJS view engine, the Express middleware, and the PostgreSQL pg packages. Note: This REST API needs only to GET /SELECT data from your database. Required packages include pg, express, and ejs.
11. Create all node.js and ejs files like the example code created during the lectures. One file should implement the routes for the API, another the data access layer (DAL) accessing the database view you have created, and other ejs files providing the user interface.
12. Execute your Node.js application and test the application via a localhost http request. Iterate your development until your website service successfully displays the database view query results to the web page.
13. Be sure to save files to the respective QAP3 project folders.

**Approach:**

I suggest you work on this QAP in a few phases.

Phase 1: determine the data requirements and how all the data can be queried from one SELECT statement implemented as a database VIEW.

Phase 2: using npm start a new project, install the required npm modules, and create a project folder structure that reflects a well designed full-stack application.

Phase 3: Build the index.js file which implements the express code to serve up a web server. At this time, in your routes folder, also create the js file(s) which define the required route(s).

Phase 4: in your routes file(s) create mock data arrays which implement the data formatted the same as would be returned from the data access layer. Review the lecture sample code for an example.

Phase 5. Build the html files required to display the data. Test the display of the pure HTML before beginning to embed ejs syntax into the html. Iterate and test the ejs file with the working mock data array.

Phase 6. Build the dal service layer to retrieve data from the database VIEW. Replace the mock data array with the dal layer database function call. Continue your testing. Review the lecture sample code if required.

**Deliverables:**

1. All the CREATE statements for the PostgreSQL database VIEWs, data creation or alteration, and any table(s) changes.
2. INSERT and / or UPDATE statements required for any additional data.
3. The node.js and ejs files used to implement all the Model-View-Controller and the Data Access Layer (DAL) using the respective database VIEW.
4. Be sure to include your package.json file

**Project Submission:**

All your project files should be saved to a single working directory (folder) with sub-folders and an easily understood file naming convention. All files should be added to a single zip file using your name as part of the file name. (eg. PeterRawsthorneQAP1.zip). Include **only** your project files in the zip file. DO NOT include other NPM files or folders as part of your project. Do not include the **node\_modules** folder. Your QAP will **not** be considered if it contains any additional files as this can make the project zip file > 30 MB in size!