Development Technical Test

Overview

This test gives candidates an opportunity to demonstrate their ability to take some simple requirements and turn them into working, production-grade code.

Our preference for the technical stack to use for the test, which is based on our project's actual technical stack, is:

1. AngularJS + Bootstrap responsive framework for the web UI
2. NodeJS for the REST API (with Express or Koa).
3. MySQL (or MariaDB) for database

If you are not comfortable using NodeJS, you may use a different stack (e.g. Python, PHP, .Net, Java) for the backend.
If you are not comfortable using MySQL/MariaDB, you may use different relational database or a NoSQL solution (e.g. RethinkDB).
The exercise's requirements will be more simplified than our "real" needs, however the code you write needs to be written as if it was for real world production use - so needs to perform well, be
easy to maintain and extend, and be robust. The code needs to reflect what you believe are good development practices.
Once your exercise is complete, email it to us in a ZIP file (including the source code for your application and the SQL scripts to create any database objects) and we will assess it. Include a "readme" file that explains what languages/frameworks are used, any assumptions you made, any information we need on how to build and run the code.

The test is to be completed on your own time. You will need to complete the exercise within 7 days. If you will not be able to finish within 7 days, you need to get in touch with us ahead of time to discuss options.

Make sure that your write all the code, and the code is a reflection of your own design choices - don't get someone else to write it for you, or do your thinking, or borrow code from other places. You will need to be able to explain your design choices - what you did and why - and extend on them.

If we think your exercise looks promising, we will then get you to come in and do a 1-2 hour Pairing Session with one of our Architects. The architect will get you to explain your design choices, any assumptions you made (where there was ambiguity), and get you to make some changes to your code on the spot to meet some new/changed requirements.

For the pairing session you will need to bring in your own laptop with any tools you need (e.g. IDE, Database Engine, frameworks etc) already installed. If you cannot provide your own laptop you will need to discuss alternatives with us ahead of time.

As a stretch goal, include some automated testing. You don't need to test everything - use your own judgement on what is most important to test, and how.

Exercises

Step 1. Creating the database

In this step, you will create a database we can use to retrieve and store data.

Consider the following statements:

1. The system will store many Products, from many Brands.
2. Each Product belongs to a Brand. Each Brand can have many Products.
3. Products need to have an ID, Product Name, Description, Price, Colour, Date Created and Availability Status (e.g. In Stock, Out of Stock, Archived).
4. A Brand needs to have an ID, Name and Description.
5. A Product can have Reviews. Each Review is written by a User.
6. A Review needs to have an ID, Rating, Comment. The rating can only be a number between 0 to 10.
7. A User needs to have an ID, User Type, User Name, Email and Date of Birth. Email should be unique (i.e. two users can't have the same email). User Types will be Customer and Merchant

Create database tables that can contain the data structure described above. Use your own judgement on which exact tables/fields are required (naming conventions, data types, object relationships etc).

Create script(s) to set up some dummy data we can use for querying this database.

Step 2. Creating the public REST API

In this step, we will expose web services that will use the database you created in the previous step.

Write the code needed to implement these user stories:

As any user, I want to call a web service to find the latest Products, so I can display them.

Acceptance Criteria:

1. Returns the 10 newest products.
2. For each product, needs to include the Product ID, Name, Description, Brand Name and the most recent Review for this product (including the User Name and Review Summary).
3. Accepts an optional Brand ID parameter to filter the results. If a Brand ID provided, returns only products for that brand.

Technical acceptance criteria:

1. RESTful web service that uses JSON as message format
2. No authentication required

As a Customer, I want to call a web service to Create a Review, so I can give feedback on a product.

Acceptance Criteria:

1. Accepts parameters for User ID, Product ID, Rating, Comment
2. User ID must exist, and must be for a user with type "Customer"
3. Should perform basic validation and return a descriptive error message if validation fails

Technical acceptance criteria:

1. RESTful web service that uses JSON as message format
2. No authentication required

Step 3. Create the "View Product List" web page

In this step you will call the web services you created in the previous step. You can choose to do this either:

1. Server side, where you will make a server-side request to the web service and server-side render a page showing the results
2. Client side, where you will use JS/AJAX to make a request to the web service and render the results in the browser.

Write the code needed to implement this user story:

As a Customer, I want to view the latest Products on the website, so I can choose what to buy.

Acceptance Criteria:

1. The page accepts an optional Brand ID parameter on the URL to filter the results. If a Brand ID provided, displays only products for that brand. If no Brand ID, then show products from any Brand.
2. If no products found, displays a friendly message to the user.
3. If matching products are found, the page will display a list of the (up to) 10 newest products, including their Product Name & Description, Brand Name and the most recent review (including the User Name, Rating and Comment).
4. Each product in the list should have an "Add Review" link that will take the user to a different page where they can add a Review (this page to be created in the next step).
5. The page should use at least some basic visual styling, but it's up to you how far you want to take this (depending on your front-end development skills etc).

Technical acceptance criteria:

1. Needs to call the Find Products web service you previously created
2. Needs to work in current versions of Chrome and Firefox
3. No authentication required

Step 4. Create the "Add Review" web page

In this step you will create a web page that allows customers to add a new review for a product. You can choose to either do this AJAX style or just with simple page posts, depending on your skills.

Write the code needed to implement this user story:

As a Customer, I want to add a new Review via the website, so I can give feedback on a product.

Acceptance criteria:

1. The page accepts a Product ID on the URL. This product must exist.
2. If no matching product found, then show a descriptive message to the user.
3. If the product does exist, then show the Product Name on the form (read only).
4. The user can use a form to enter their Email, Rating and Comment. They can then click a "Save" button to submit their review.
5. If the review fails validation, the user should see a descriptive message on the page.
6. If the save is successful, the user should be redirected back to the View Product List page
7. The page should have some basic visual styling.

Technical acceptance criteria:

1. Needs to call the Add Review web service you previously created
2. Needs to work in current versions of Chrome and Firefox
3. No authentication required