**Vue Certification mini-project **

# **GOAL: Build a small car configurator application that allows the user to select different car options**

The app you have to build is a simplified version of <https://www.tesla.com/modelx/design>. You can use that website for inspiration if you want, but our API and possible configurations are a lot simpler. You can see a [video demo of the expected app here](https://youtu.be/kxvYtLLKv8g).

For this project, we want you to start from the code repository: https://rxgit.radixweb.in/rxprojects/rxpractice/php/vue-project This will make it easier to start coding as the setup is already done and the API is included.

Once your code is completed, we recommend using Vercel or Netlify to host the project and publish a built version of your code.

To submit your work, you’ll need to provide the link to theGit repository that contains your code, along with a **public URL** to test the app in a browser.

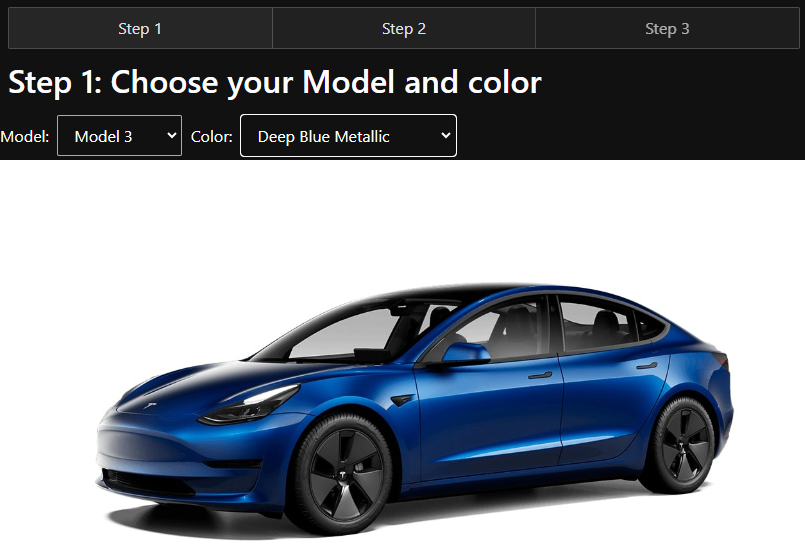
**Important rules and notes - please read carefully:**

You have to **write the code yourself**. Submitting code already submitted by a friend or colleague is not allowed and will result in disqualification from the certification exam.

Code **quality and best practices matter**. Please do not use the **any** type in Typescript or have a single component in your application. Your code **MUST** use proper types for all variables, methods, parameters, etc. Failure to do so will automatically disqualify your submission.

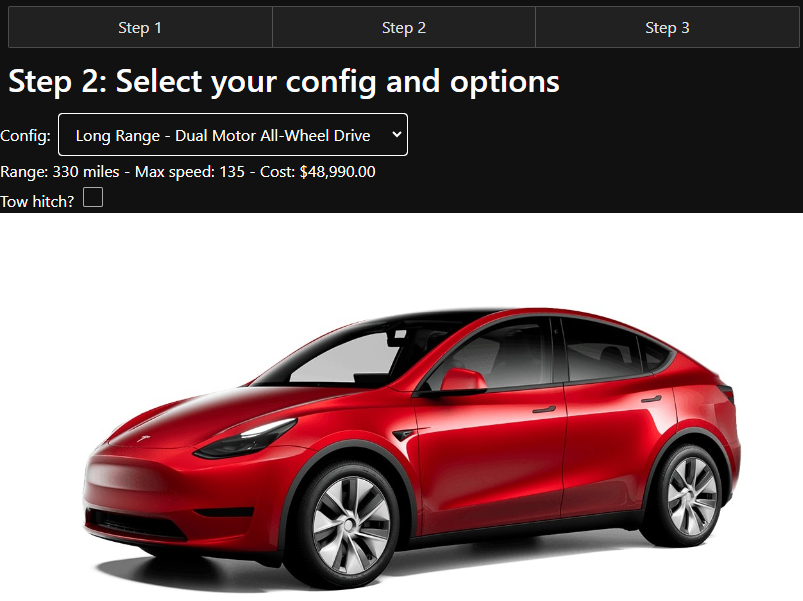
It should go without saying that **your application will be disqualified and your certification exam marked as failed if the application has bugs, doesn’t implement all of the features, or doesn’t follow the instructions of this document.**  
  
**Finally**, once your application is working, fully tested, and follows best practices (using types and proper component architecture), **submit your work**

**STEP #1 - Implement the ability to select a Tesla car model and color**

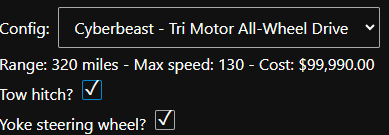


* Models and colors must be retrieved from the API included in the project and accessible at the **/models** endpoint.
* Images for all models and colors can be found at <https://interstate21.com/tesla-app/images/>
* When a **Model** and **Color** are selected, the proper image should be displayed on the screen as illustrated above.
* Step 2 navigation at the top of the screen must be disabled as long as the user hasn’t selected a model and color. Use the **Vue router** to implement the 3 steps.
* Note that styling is not that important for this project, and the layout of your application can be different as long as the application works properly. This is an Vue certification, not a CSS or HTML certification.
* The two dropdowns must have an HTML attribute ID equal to **modelSelect** and **colorSelect**. The step buttons must have IDs of **step1**, **step2**, and **step3** respectively.

**STEP #2 - Allow the user to select a car config and options**

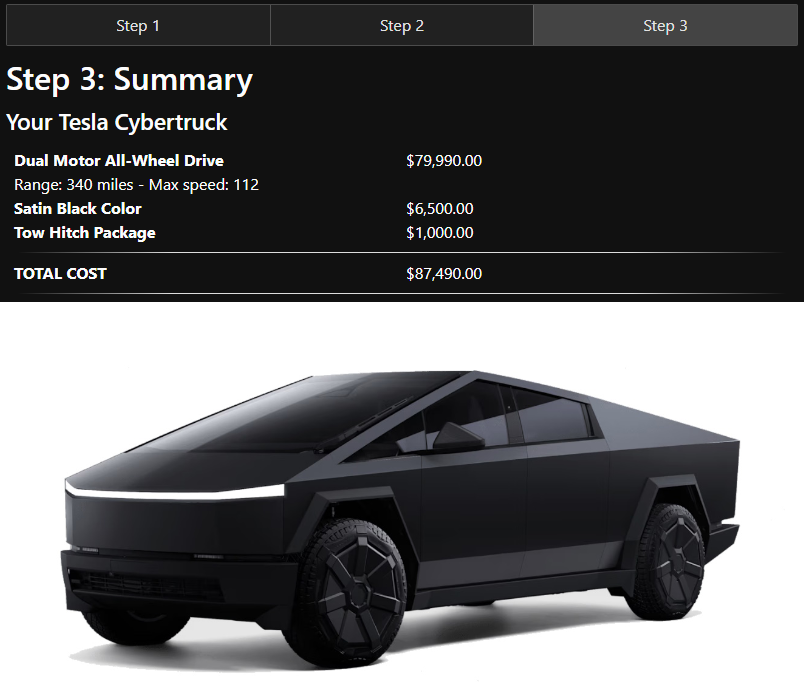


* In the step 2 screen, use the **/options/:modelCode** API endpoint to get the different configs and options available for the selected car.
* Models can have different configs with different prices. Two other options cost $1,000 and **must only be displayed (not checked - see image above) when available** on the selected car model: **yoke** [steering wheel](https://www.motortrend.com/reviews/2022-tesla-model-s-plaid-steering-yoke-wheel-review/) and **tow hitch** package. The API has two booleans to indicate whether these options are available or not on the select model.
* When a config is selected, display the associated range, max speed, and cost. For example:



* The dropdowns must have an HTML ID equal to **configSelect,** and if tow hitch or yoke are available, those checkboxes must have an ID of **includeYoke** and **includeTow.**

**STEP #3 - Display a recap of the total cost of the selected car model and options**



* Step 3 should only be accessible when the user has selected a config in step 2.
* Display the cost of every chosen option (color, config, yoke, tow hitch, etc.) in properly formatted USD prices as illustrated above, as well as the total cost.
* The user should be able to go back to step 1 or 2 and change the configuration, then come back to step 3 to see the updated cost. Use [this video as a reference if needed](https://youtu.be/kxvYtLLKv8g).