# Node.js Exam – Car Ad System

## Route GET / (5 Points)

Create a Node.js web server which is capable of returning an “index.html” file containing a welcome message and menu. Add all available routes from below to the menu. On this page visualize 6 cars ads, sorted by price.

## Route GET /create (5 Points)

The server should be able to return a simple form for creating a car. Each car should have Make, Model and Price. The form should have three inputs – one for the make, one for the model and one for the price.

## Route POST /create (20 Points)

When the form is submitted, the server should set some unique ID to the car and save it on some “database”. Database can be a simple in memory array of objects or a JSON file. All fields should be validated and should not be empty. You can process the invalid input in whatever way you see fit (a simple error message is more than enough).

## Route GET /all (10 Points)

When this route is reached the server should return dynamically generated HTML containing list of all cars with their make, model, price, date of creation and number of total views. Deleted cars should not be shown in the list. Cars should be listed sorted by their date of creation in ascending order. Each cars should have a link to “/details/{id}” where “id” is the ID of the car. Links are written with <a href=”url”>My link</a>.

## Route GET /details/{id} (10 Points)

When this route is reached, details about the car with the provided ID should be showed. When this URL is open a view should be counted on the car with the provided ID. The make, the model and the price views should be displayed. A button named “DELETE” should exist on the page. When clicked, the server should mark the car as deleted.

## Add undelete option (5 Points)

Allow the car to be undeleted. Change the button text and behavior depending on the state of the car. Deleted cars can be reached only by the URL.

## Add form for comments on the car details (5 Points)

Add form to the car details allowing the user to save comments for the current car. The form should have two fields - for username and for the comment text.

## Route POST/details/{id}/comment (15 Points)

The form for comments should save the data on the above route. The comment should be saved for the car with the provided ID. Validate the comment. It should not be empty. Save the date of the comment too. Add on the “/details/{id}” page all comments made for the corresponding car.

## Add image upload for each car (20 Points)

Add an input type “file” to the “/create” form allowing for the user to save an image for each car. Save the file on a server folder named “images”. Change the file name so that there will not be any collisions if the user sends two files with the same name. Show the file on the car details page. Use <img src=”imageSrc” /> HTML tag.

## Add statistics page (5 Points)

Add GET /stats route which is available only if “My-Authorization” header with “Admin” value is provided in the request (otherwise return 404). Show all cars listed (including the deleted ones), the total number of comments and total number of views.

**OTHER REQUIREMENTS**

* Use Node.js as web server
* Add initial sample data for easier testing of the application
* Follow the standard or other linter - <https://github.com/feross/standard>
* You may use whatever frameworks you like (including express.js)