# Spring Data

# Car Repair Service

[Link for Car Repair Service Exam in Judge](https://judge.softuni.org/Contests/4238/Java-DB-Spring-Data-Exam-TEST-03-December-2022)

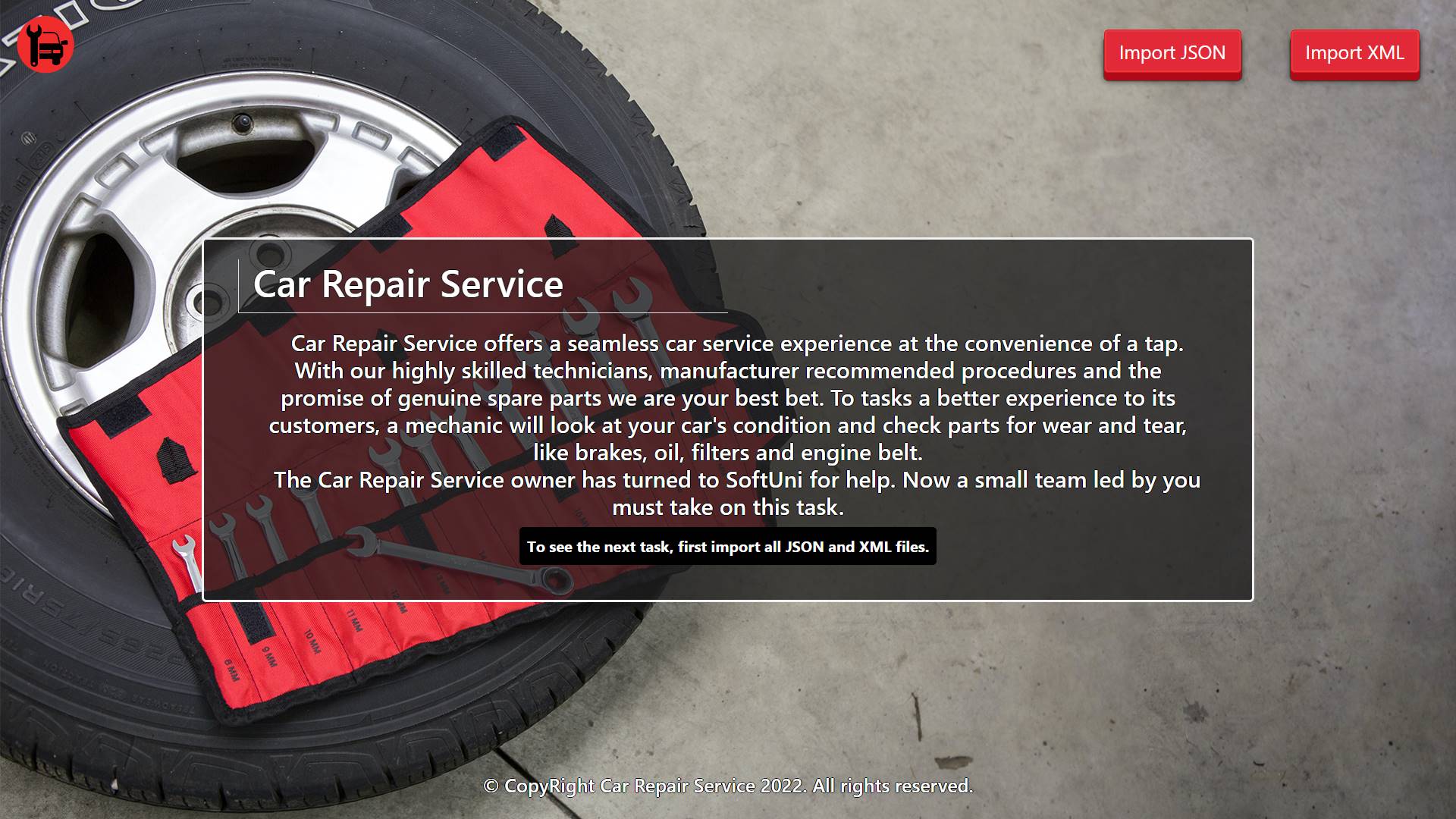
Car Repair Service offers a seamless car service experience at the convenience of a tap. With our highly skilled technicians, manufacturer recommended procedures and the promise of genuine spare parts we are your best bet. To tasks a better experience to its customers, a mechanic will look at your car's condition and check parts for wear and tear, like brakes, oil, filters and engine belt. The Car Repair Service owner has turned to SoftUni for help. Now a small team led by you must take on this task.

## Functionality Overview

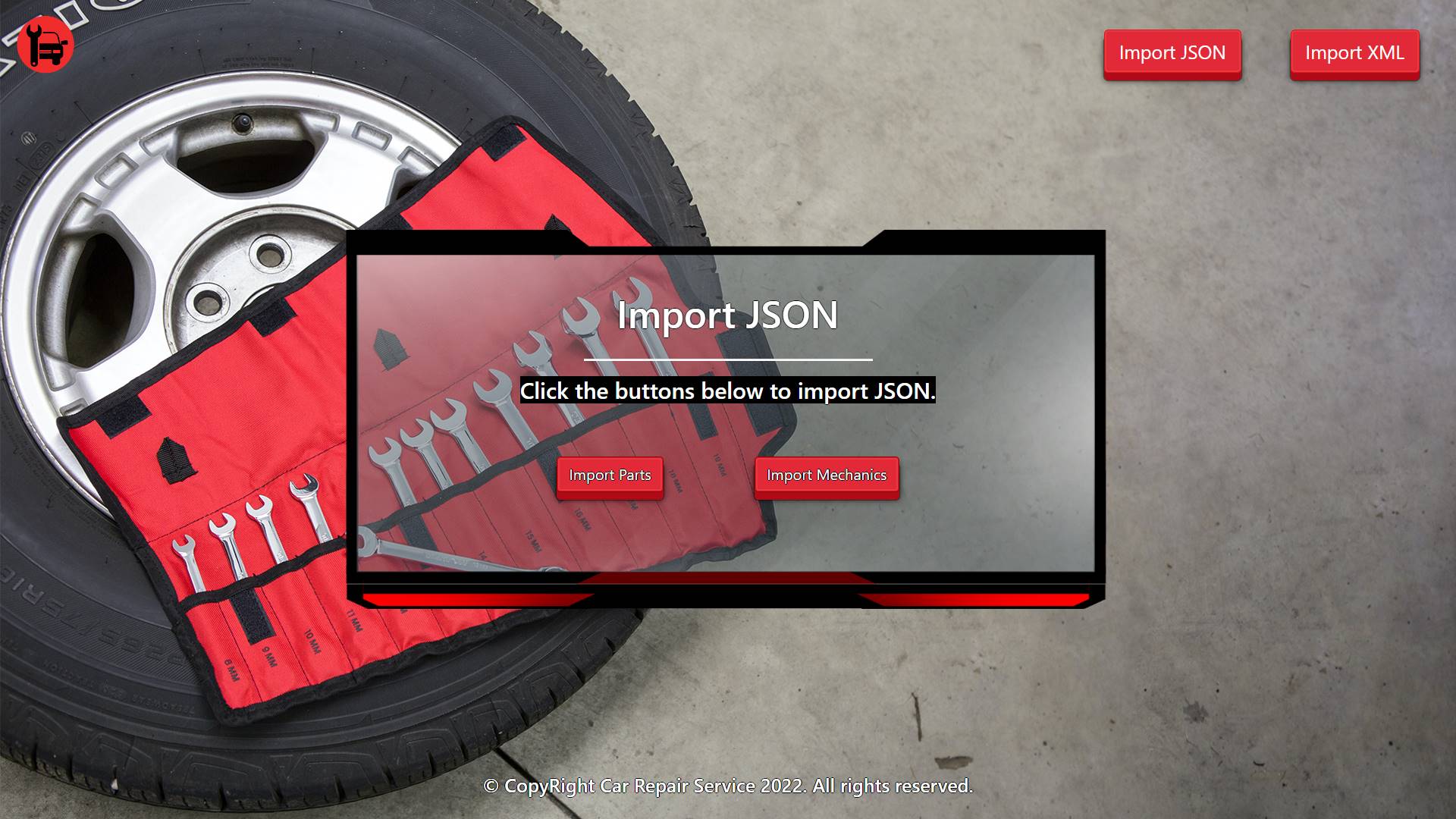
The application should be able to easily **import** hard-formatted data and **support functionalities** for also **exporting** the imported data. The application is called – **Car Repair Service**.

Look at the pictures below to see what must happen:

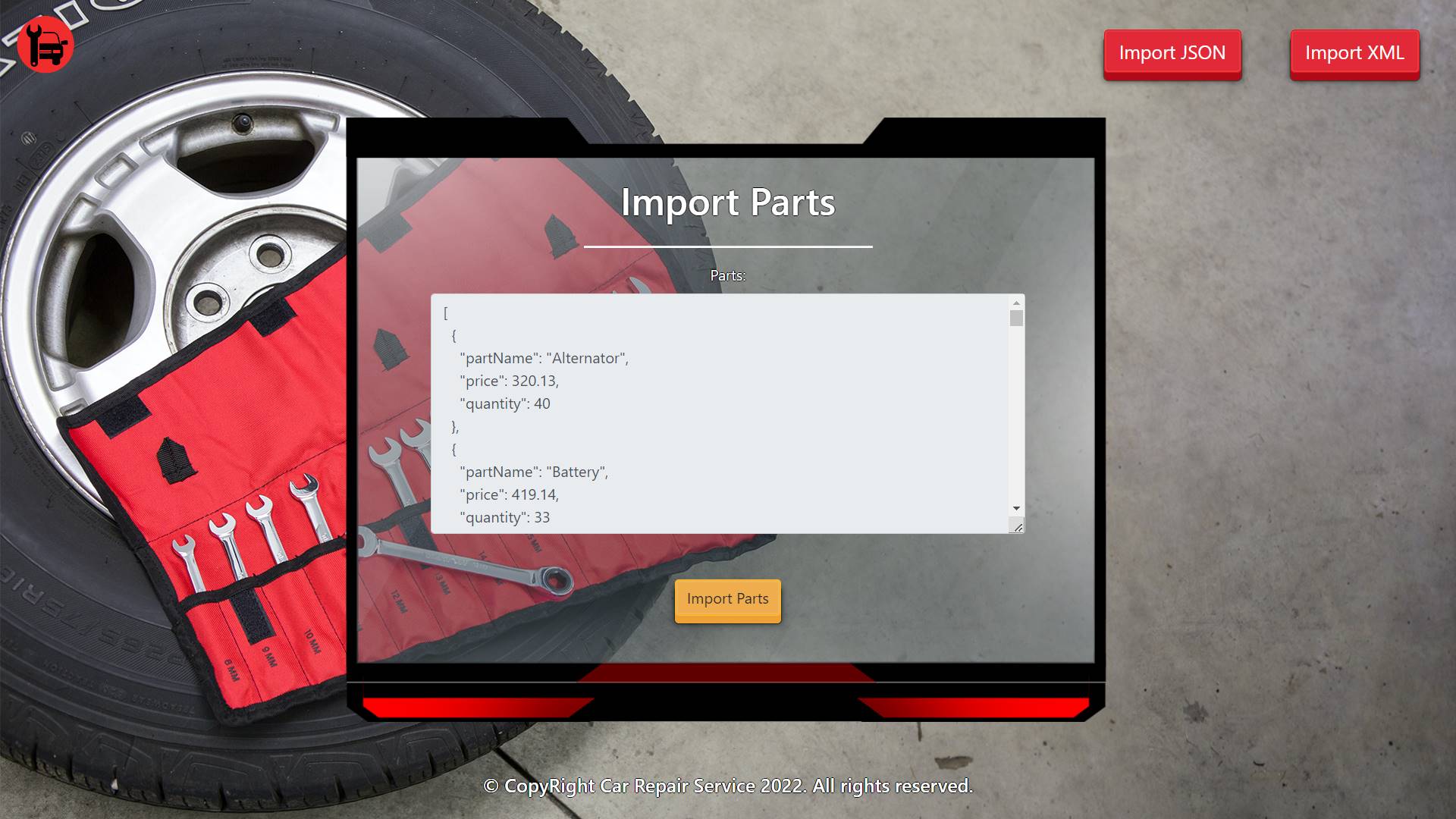
* The home page before importing anything:



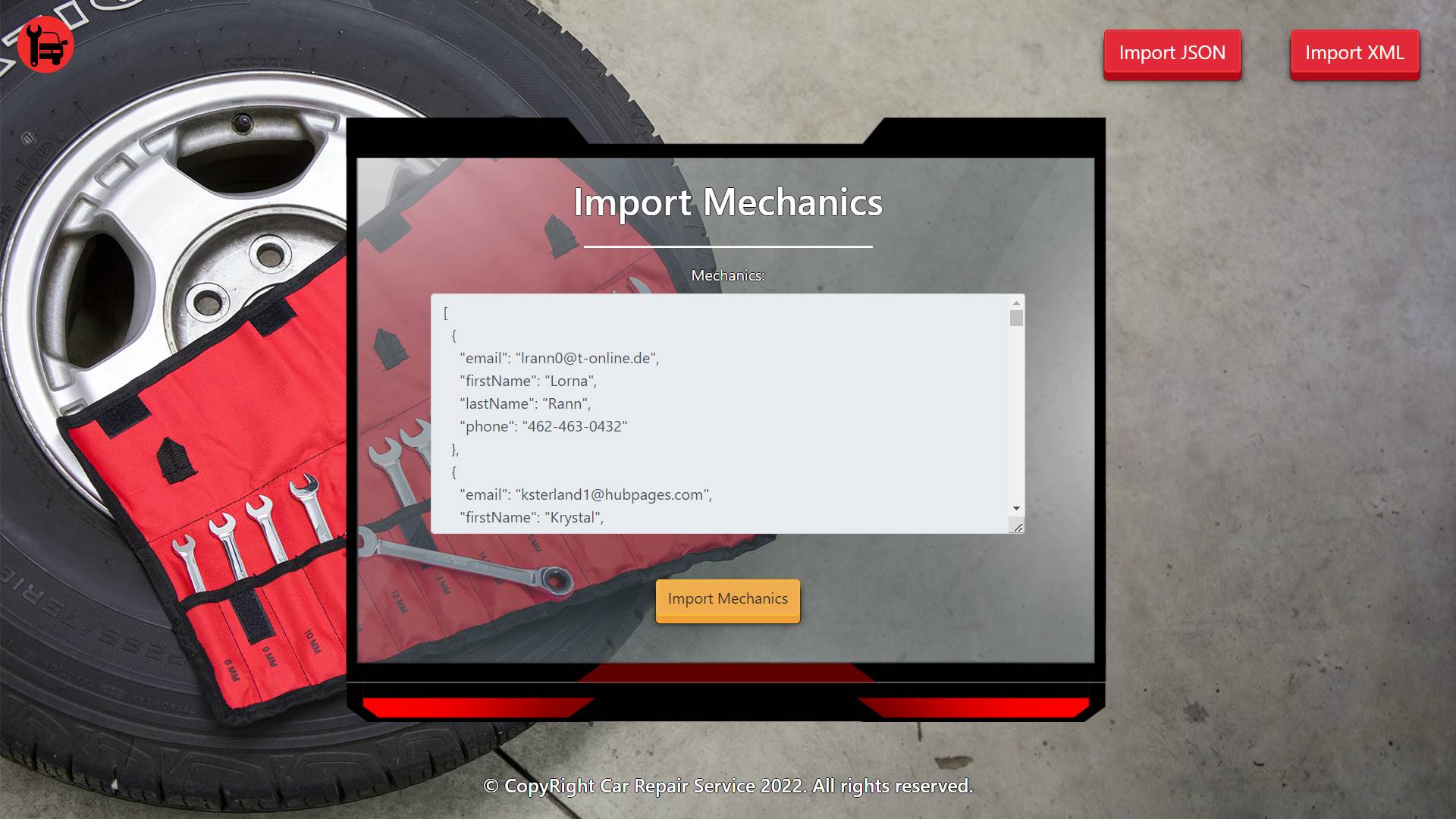
* The import JSON page before importing anything:



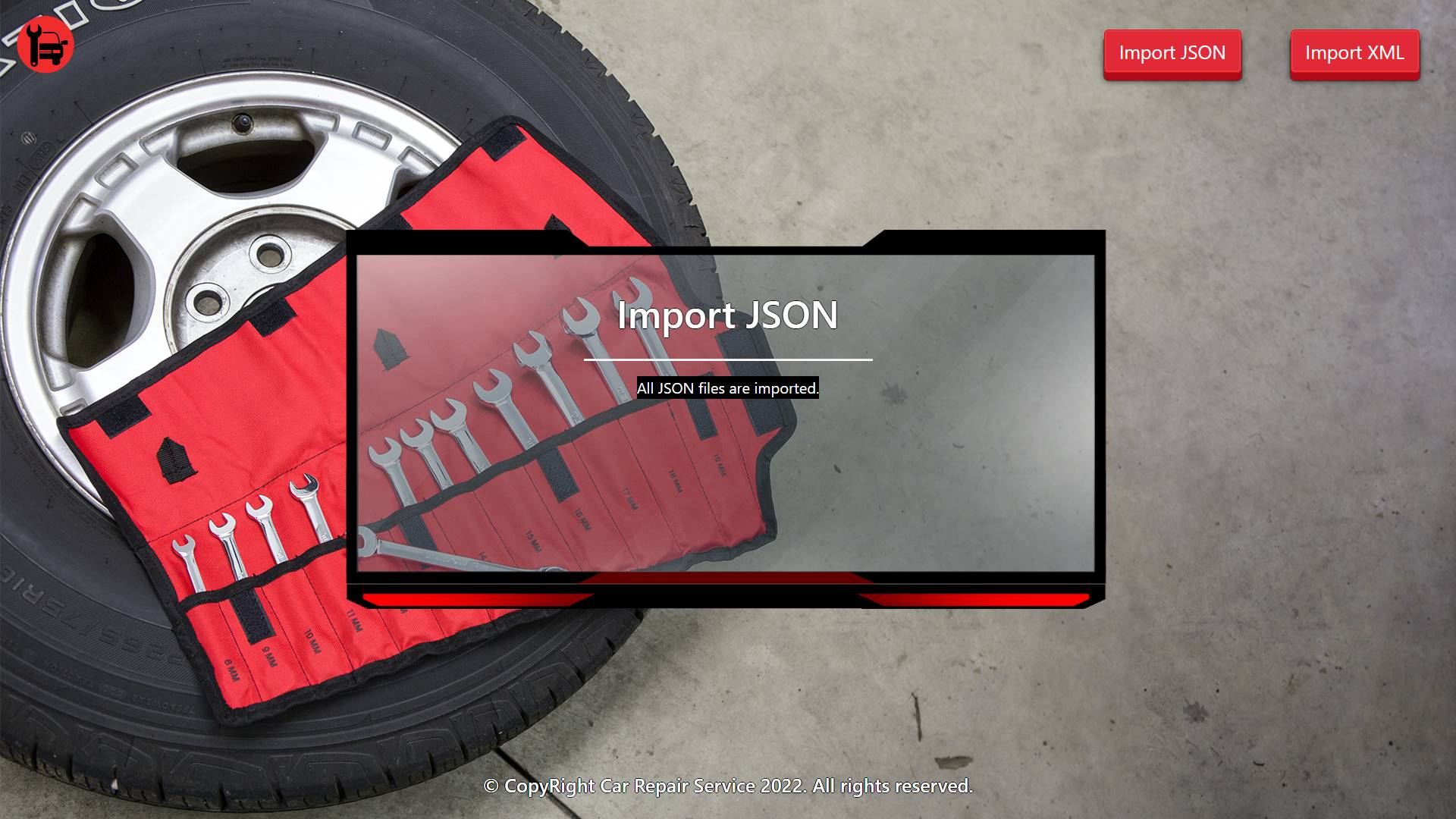
* Import the parts first:



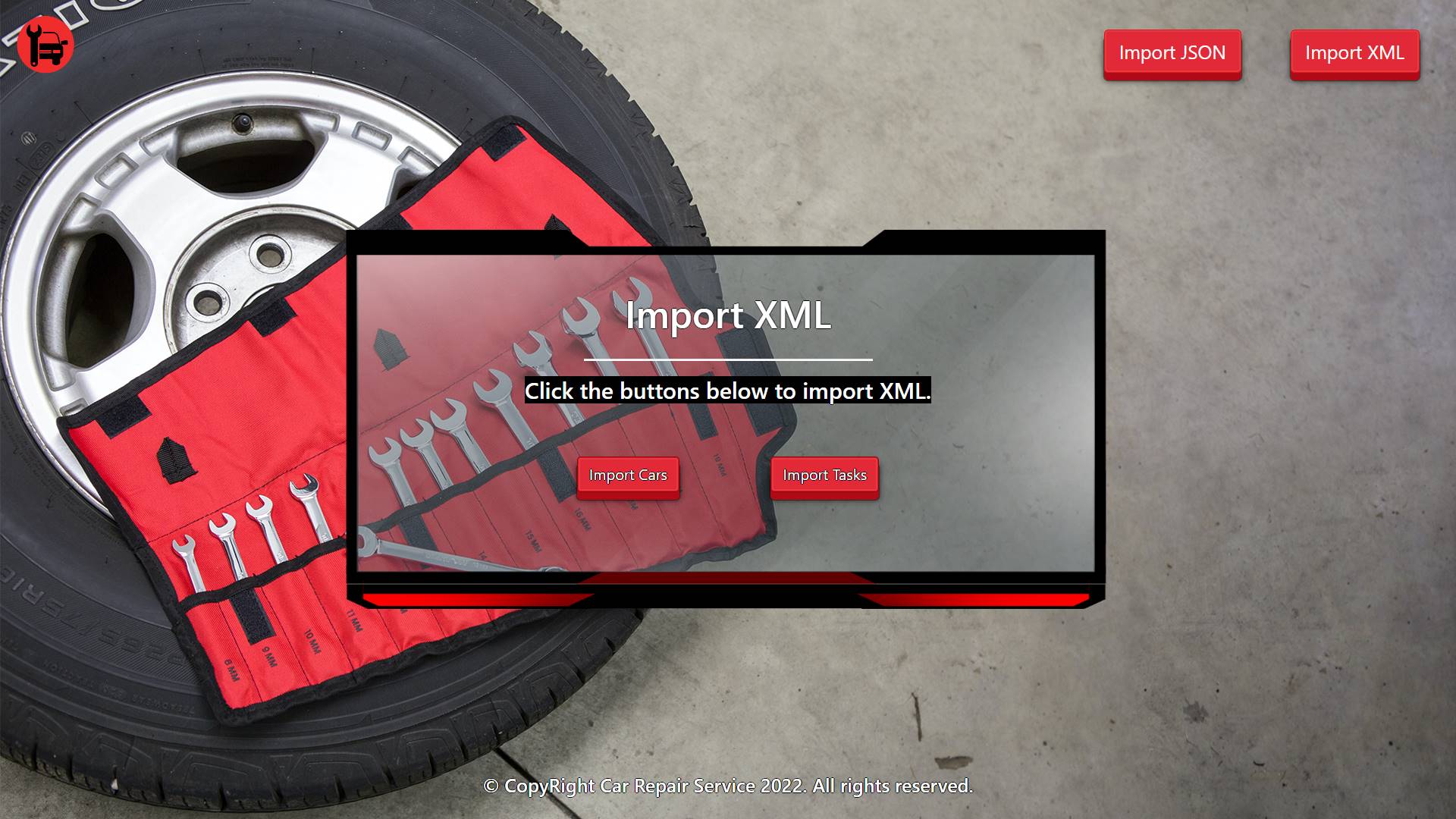
* Import the mechanics second:



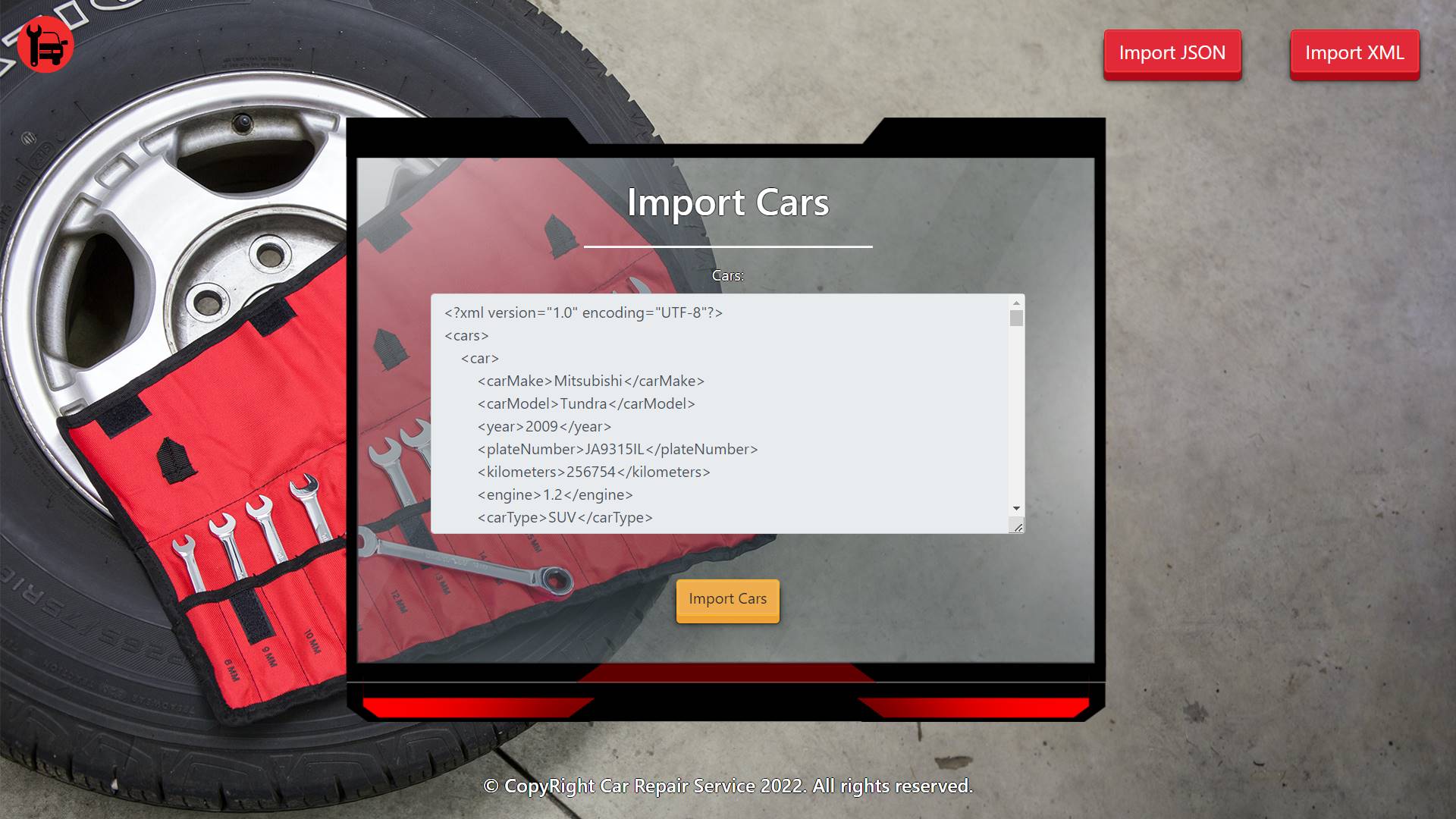
* The import JSON page after importing both files:



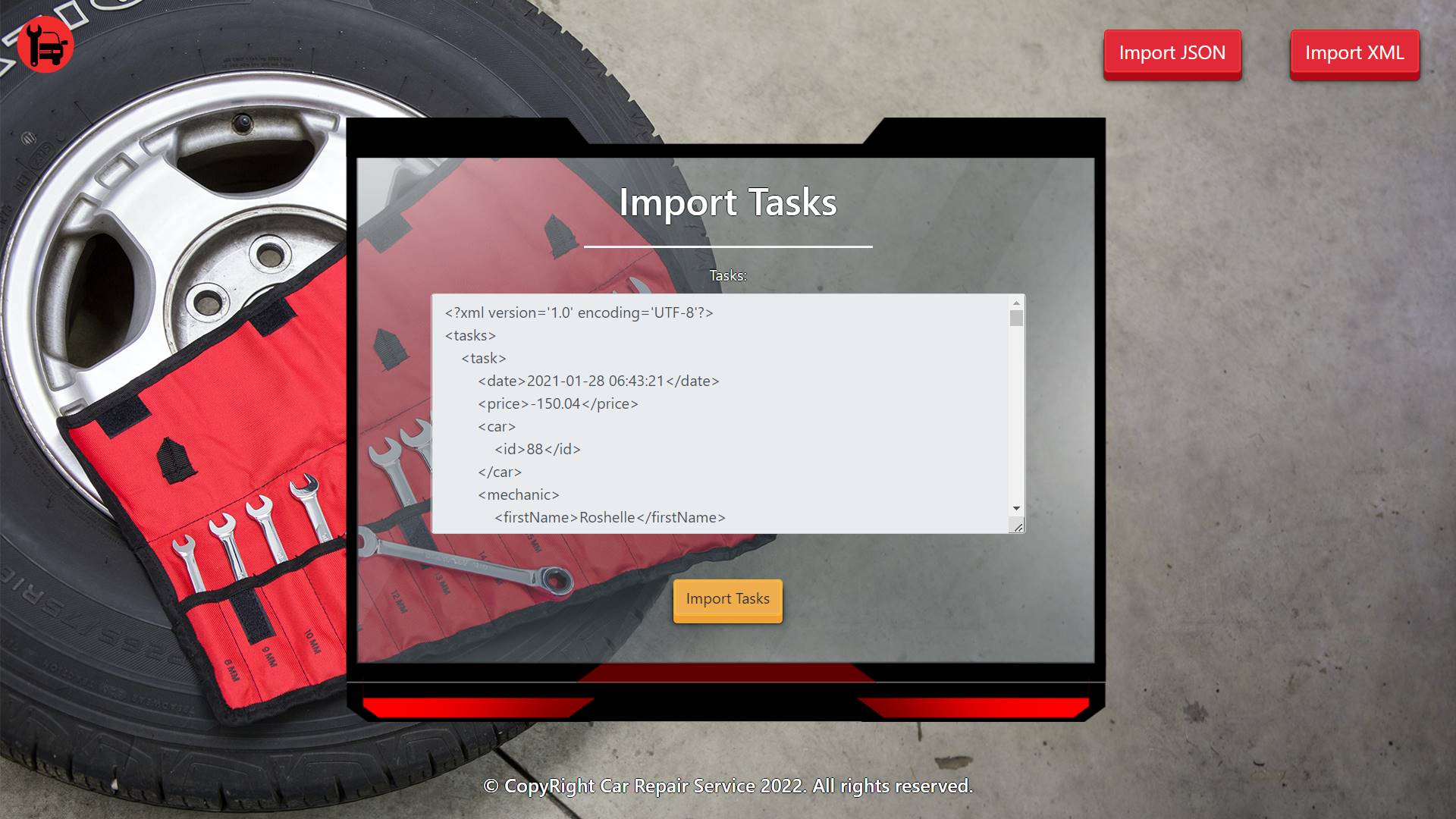
* The import XML page before importing the given data:



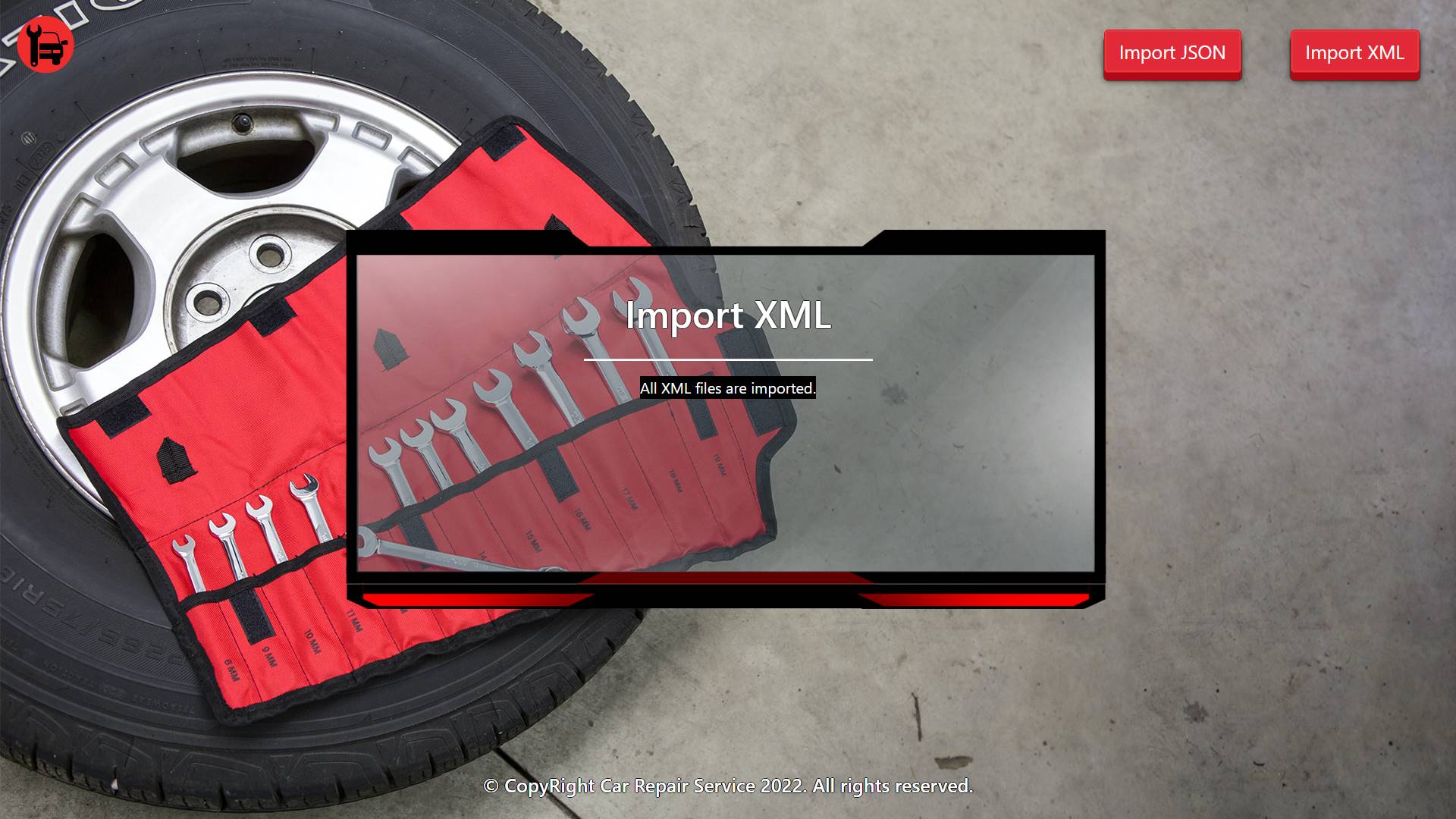
* Import the cars data:



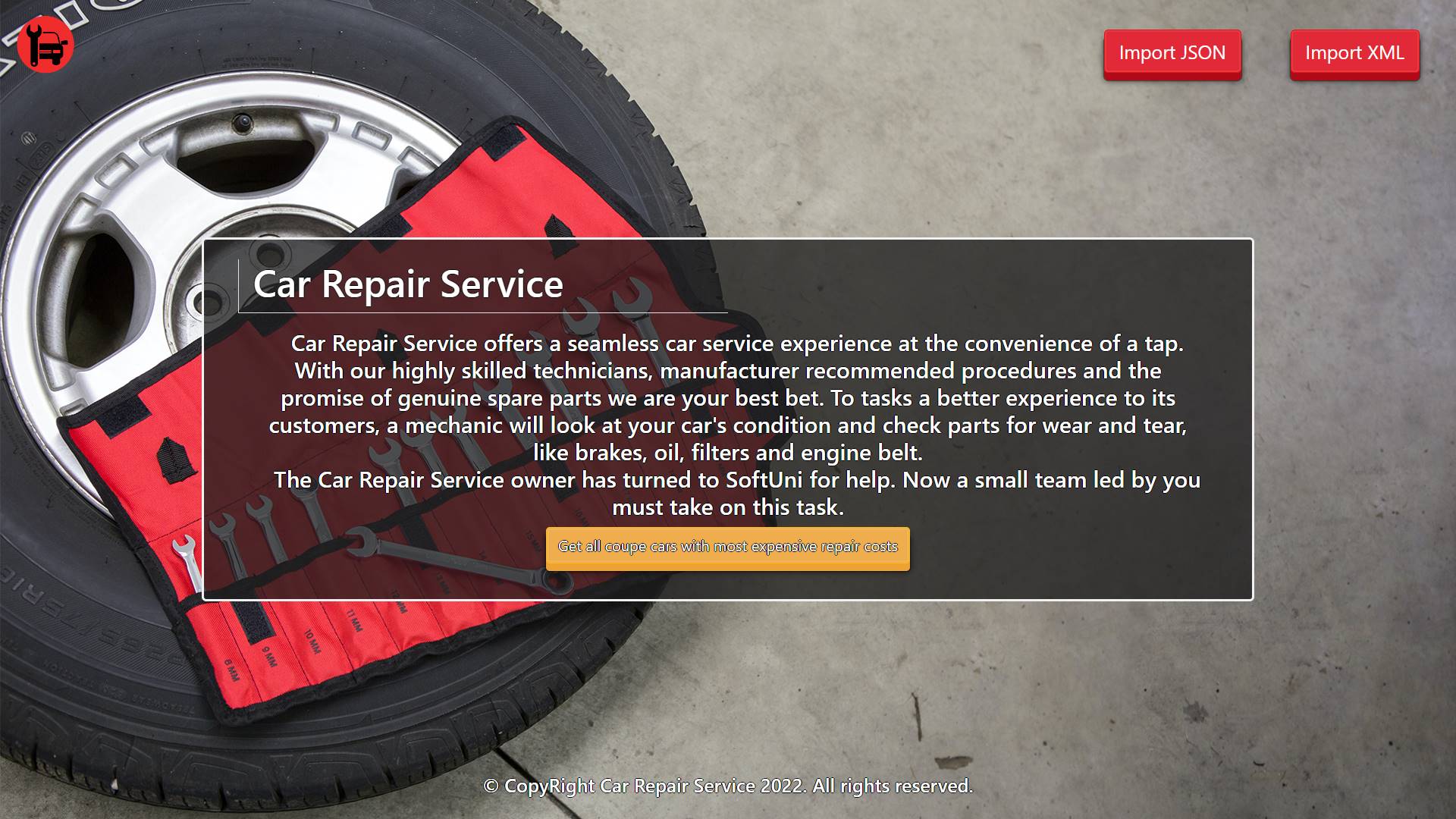
* Import the tasks data:



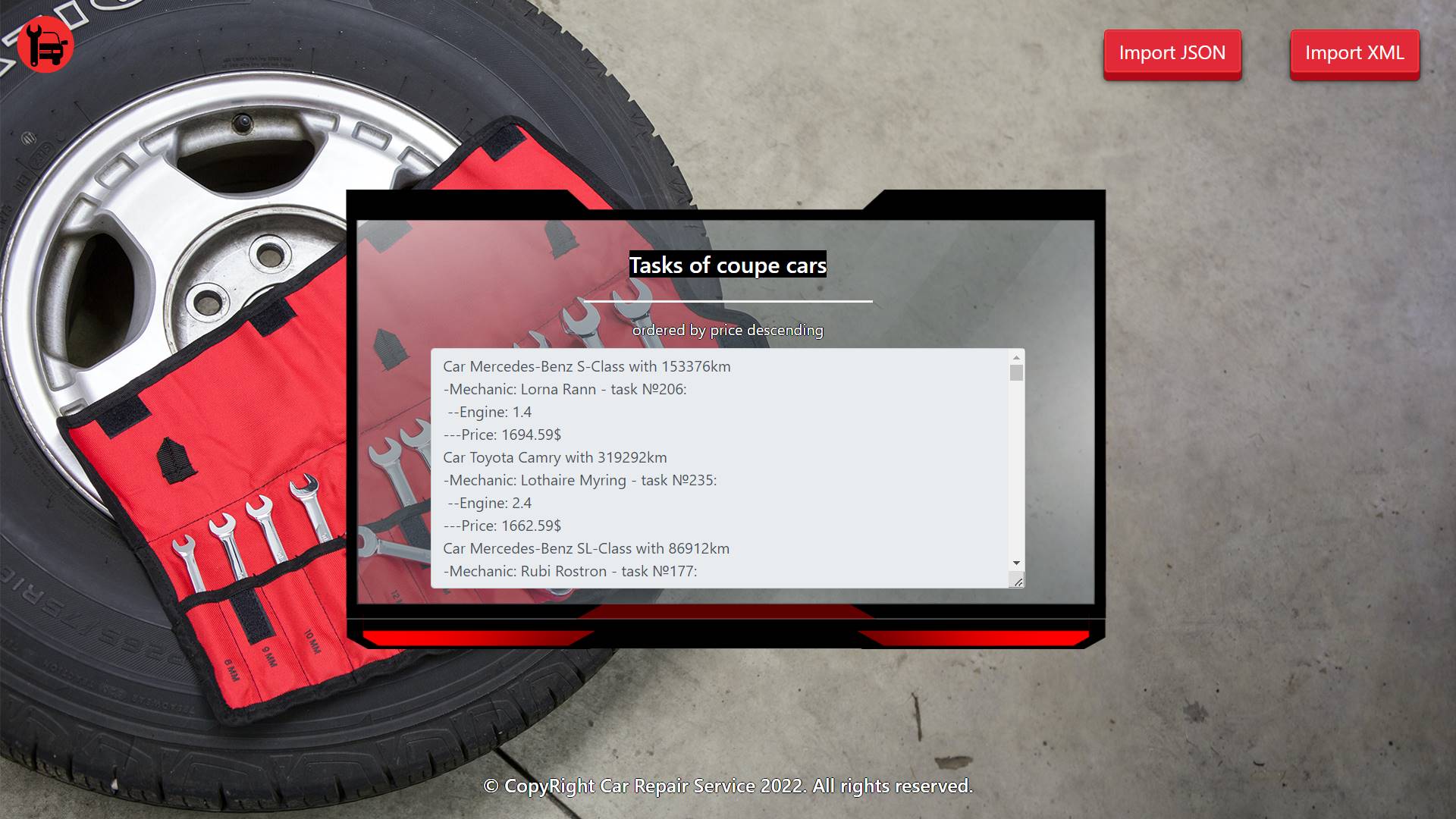
* The import XML page after importing the data:



* The home page after the data is imported:



* Export tasks for coupe cars:



## Project Skeleton Overview

You will be given a **skeleton**, containing a **certain architecture (MVC)** with **several classes**, some of which are completely empty. The **Skeleton** will include the **files** with which you will **seed** the **database**.

*When submitting your project in Judge please make a* ***.zip*** *file only with* ***src*** *folder and* ***pom.xml*** *file.  
There are four problems in* ***Judge****:*

***1. Database*** *will test the structure of your database.*

***2. Service*** *will test the two methods* ***areImported()*** *and* ***readClassFromFile()*** *in every service class.****3. Imports*** *will test the whole logic of your* ***importClass()*** *method in every service class.*

***4. Export*** *will test the logic of the final* ***export*** *button which is in the* ***ExportController****.*

*Please note that the automated system Judge may take up to 5 minutes to show results for each problem. We kindly ask you to be patient.*

## Model Definition

There are 4 main models that the **Car Repair Service database** application should contain in its functionality.

Design them in the **most appropriate** way, considering the following **data constraints**:

### Part

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* part name – accepts **char sequence** (between **2** to **19** inclusive). The values are **unique in the database**.
* price – a **floating point number**. Must be between **10** and **2000** (both numbers are **INCLUSIVE**)**.**
* quantity – accepts a **positive number**.

### Mechanic

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* first name – accepts **char sequences** as valueswhere their character length value **higher than or equal to 2.** The values are **unique in the database**.
* last name – accepts **char sequences** as valueswhere their character length value **higher than or equal to 2.**
* email – an **email –** (must contains ‘@’ and ‘.’ – dot). The email of a seller is **unique**.
* phone – accepts **char sequences** as valueswhere their character length value **higher than or equal to 2.** Can be **nullable.** The values are **unique in the database**.

### Car

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* car type – the enumeration, one of the following – **SUV, coupe, sport**
* car make – accepts **char** **sequence** (between **2** to **30** inclusive).
* car model – accepts **char** **sequence** (between **2** to **30** inclusive).
* year – accepts a **positive number**.
* plate number – accepts **char** **sequence** (between **2** to **30** inclusive). The values are **unique in the database**.
* kilometers – accepts a **positive number**.
* engine – accepts **number** values that are **more** than or **equal** to **1.00.**

### Task

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* price – accepts a very big **positive number**.
* **date** – a **date** and **time** of registering the task in the "**yyyy-MM-dd HH:mm:ss**" format.
* **Constraint**: The task table has a relation with the parts table.
* **Constraint**: The task table has a relation with the mechanics table.
* **Constraint**: The task table has a relation with the cars table.

#### Relationships

Your partners gave you a little hint about the more complex relationships in the database, so that you can implement it correctly.

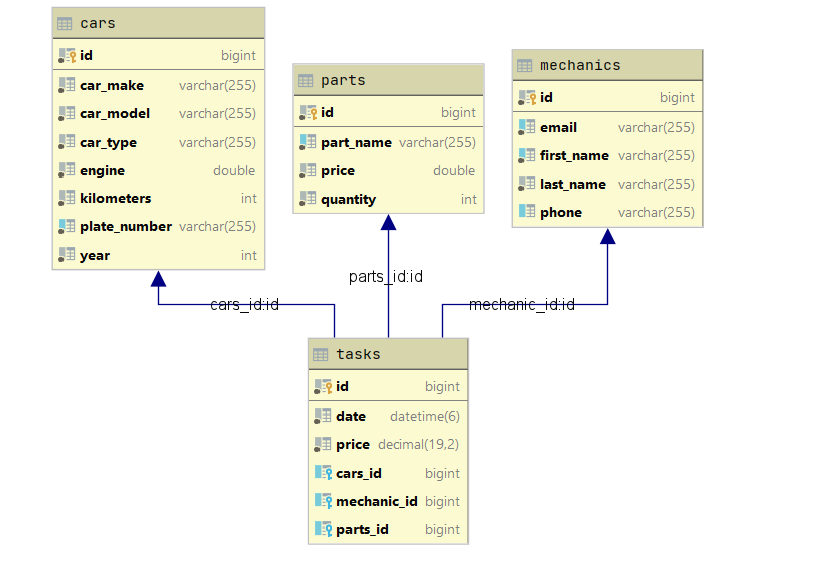
One **Task** may have only one **Mechanic**, but one **Mechanic** may have many **Tasks**.

One **Task** may need only one **Part**, but one **Part** may be used in many **Tasks**.

One **Task** may have only one **Car**, but one **Car** can be in many **Tasks**.

**Constraint**:

* Name the entities and their class members **exactly** in the **format stated** above.
* All fields are **NOT NULL** unless explicitly stated to be nullable.



## Data Import

Use the provided files to populate the database with data. Import all the information from those files into the database.

**You are not allowed to modify the provided files.**

**ANY INCORRECT** data should be **ignored** and a message:

"**Invalid {part / mechanic / car / task}**" should be printed.

**When the import is finished:**

"**Successfully imported {part / mechanic / car / task} {name – price**(to second digit after decimal point) **/ first name last name / carMake – carModel / price}**"

### JSON Import

Your new colleagues have prepared some JSON data for you to import.

|  |
| --- |
| **Parts (parts.json)** |
| [  {  "partName": "Alternator",  "price": 320.13,  "quantity": 40  },  {  "partName": "Battery",  "price": 419.14,  "quantity": 33  },  {  "partName": "Performance Battery",  "price": 455.38,  "quantity": 15  },  {  "partName": "Alternator",  "price": 240.13,  "quantity": 18  },  {  "partName": "Battery Box",  "price": 590.43,  "quantity": 21  },  ... |
| Successfully imported part Alternator - 320.13  Successfully imported part Battery - 419.14  Successfully imported part Performance Battery - 455.38  Invalid part  Successfully imported part Battery Box - 590.43  Invalid part  Invalid part  Invalid part  Successfully imported part Battery Cable - 269.73 … |

**Constraint:**

* **If the mechanic with the same email already exists in the DB return "Invalid mechanic".**

|  |
| --- |
| **Mechanics (mechanics.json)** |
| [  {  "email": "lrann0@t-online.de",  "firstName": "Lorna",  "lastName": "Rann",  "phone": "462-463-0432"  },  {  "email": "ksterland1@hubpages.com",  "firstName": "Krystal",  "lastName": "Sterland",  "phone": "201-115-5358"  },  {  "email": "qle@es.com",  "firstName": "Q",  "lastName": "Beck",  "phone": "101-135-6558"  },  {  "email": "wrongNumber@no.com",  "firstName": "Qentin",  "lastName": "Jock",  "phone": "1"  },  {  "email": "gledeker2@amazon.com",  "firstName": "Gerladina",  "lastName": "Ledeker",  "phone": "534-162-5734"  },  {  "email": "ksterland1@hubpages.com",  "firstName": "Deidre",  "lastName": "Goodinge",  "phone": "148-743-0753"  },  {  "email": "dhebard4@histats.com",  "firstName": "Dulcy",  "lastName": "Hebard",  "phone": "654-577-5713"  },  . . . |
| Successfully imported mechanic Lorna Rann  Successfully imported mechanic Krystal Sterland  Invalid mechanic  Invalid mechanic  Successfully imported mechanic Gerladina Ledeker  Invalid mechanic  Successfully imported mechanic Dulcy Hebard  … |

### XML Import

Your new colleagues have prepared some XML data for you to import.

**Constraint:**

* **If the car with the same plate number already exists in the DB return "Invalid car".**

**Cars (cars.xml)**

|  |
| --- |
| **Cars (cars.xml)** |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <cars>  <car>  <carMake>Mitsubishi</carMake>  <carModel>Tundra</carModel>  <year>2009</year>  <plateNumber>JA9315IL</plateNumber>  <kilometers>256754</kilometers>  <engine>1.2</engine>  <carType>SUV</carType>  </car>  <car>  <carMake>MitsubishiAventadorBmwLamborghiniOpelBentley</carMake>  <carModel>Tundra</carModel>  <year>2009</year>  <plateNumber>QA7315IL</plateNumber>  <kilometers>251754</kilometers>  <engine>1.3</engine>  <carType>SUV</carType>  </car>  <car>  <carMake>Hyundai</carMake>  <carModel>Tiburon</carModel>  <year>2006</year>  <plateNumber>JA9315IL</plateNumber>  <kilometers>18862</kilometers>  <engine>2.2</engine>  <carType>SUV</carType>  </car>  ... |
| Successfully imported car Mitsubishi - Tundra  Invalid car  Invalid car  Successfully imported car Land Rover - Range Rover  Successfully imported car Dodge - Intrepid  Invalid car  Successfully imported car Honda - S2000 ... |

**Tasks (tasks.xml)**

**Constraint:**

* **If the given mechanic name doesn’t already exist in the DB return "Invalid task".**
* **The provided part ids will always be valid.**
* **Format the price to the second digit after the decimal point.**

|  |
| --- |
| **Tasks (tasks.xml)** |
| *<?*xml version='1.0' encoding='UTF-8'*?>* <tasks>  <task>  <date>2021-01-28 06:43:21</date>  <price>-150.04</price>  <car>  <id>88</id>  </car>  <mechanic>  <firstName>Roshelle</firstName>  </mechanic>  <part>  <id>7</id>  </part>  </task>  <task>  <date>2021-01-28 06:43:21</date>  <price>240.04</price>  <car>  <id>88</id>  </car>  <mechanic>  <firstName>Roshelle</firstName>  </mechanic>  <part>  <id>7</id>  </part>  </task>  <task>  <date>2021-09-10 06:53:58</date>  <price>1325.82</price>  <car>  <id>35</id>  </car>  <mechanic>  <firstName>Ozzie</firstName>  </mechanic>  <part>  <id>32</id>  </part>  </task>  <task>  <date>2021-03-24 02:11:32</date>  <price>1033.98</price>  <car>  <id>16</id>  </car>  <mechanic>  <firstName>Lothaire</firstName>  </mechanic>  <part>  <id>39</id>  </part>  </task>  ... |
| Invalid task  Successfully imported task 240.04  Successfully imported task 1325.82  Successfully imported task 1033.98  Successfully imported task 1230.28  Invalid task  Successfully imported task 705.94  … |

## Data Export

Get ready to export the data you have imported in the previous task. Here you will have some complex database querying. Export the data in the formats specified below.

#### Export The Highest priced Tasks from the Data Base

* Extract from the database, the **car's make, car's model, mechanic's first name, mechanic's last name, task id, engine**, **task** **price (to second digit after decimal point)** of the **task**.
* **Filter only coupe cars and order them by the price in descending order.**
* Return the information in this format:
* **"Car {carMake} {carModel} with {kilometers}km**

**-Mechanic: {firstName} {lastName} - task №{taskId}:­­**

**--Engine: {engine}**

**---Price: {taskPrice}$**

**. . ."**