|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Method Name | HTTP Method | Parameters | Returns | Explanation |
| 1 | **Login** | POST | {"username":<*user’s username*>  "password: <*password*>} | { "Success": "Login Succeed" }  { "Failed": *failure reason*} | Login cannot pass in GET |
| 2 | **GetListCarsOrderedByCategory** | GET | - | [{  "CarSeriesId": *car’s series id*,  "Category": *car’s category*  }…] | Get the list of all the cars in the system ordering by categories.  Cars are related to the seller not to the client. Using GET because there is no information to deliver to the server or any wish to change any information |
| 3 | **Register** | POST | {  "username": <user’s username>  "password": <password>,  "FirstName": <user’s first name>,  "LastName": <user’s last name>,  "Email": <user’s Email>,  "Address": <user’s Address>,  "Country": <user’s Country>,  "Q1": <Authentication question 1>,  "Ans1": <Answer for Authentication question 1>,  "Q2": <Authentication question 2>,  "Ans2": <Answer for Authentication question 2>,  "Telephone": <phone number>  "CreditCardNumber": <Credit Card Number>,  "Categories":[ <category1>, <Category2>…<category> ]  } | { "success": "New user created" }  OR  { "Failed": *failure reason*} | Register crate a new record in the DB so the POST is the best method for that.  \*all values in the JSON should be strings (Between “” ) |
| 4 | **ForgetPassword** | POST | {“username”:<user’s username>} | [  {  "Q1": *Authentication question 1*,  "Q2*": Authentication question 2*  }  ]  Or  { "Failed": *failure reason*} | Return the authentication questions that the user set in the registration process.  Post because the username is confidential |
| 5 | **RestorePassword** | POST | {  "username:<user’s username>,  "Ans1": <Answer for Authentication question 1>,  "Ans2": <Answer for Authentication question 2>} | {"password":*user’s password*  }  Or  { "Failed": *failure reason*} | The answers are confidential so cant be transfer via GET |
| 6 | **GetPopularCars** | GET | - | [{  "CarSeriesID*":The car’s series id,*  "Sold": *num of unit that sold in the last 7 days*  }…] | Return up to 5 cars that are best seller in the last week  This is in the server side because there is an algorithm that has to decide which cars to show.  Cars are related to seller not to the client. Using GET because there is no information to deliver to the server or any wish to change any information |
| 7 | **GetNewCars** |  | - | [  {  "CarSeriesID": *carSeriesID*  …  },  ] | Return the new cars that has been asdded to the system during the last month.  Cars are related to the seller not to the client. Using GET because there is no information to deliver to the server or any wish to change any information |
| 8 | **GetRecommends** | POST | {"username":<*user’s username*>} | [  {  "CarSeriesID": *carSeriesID*  …  },  ] | Return recommends of car that fits to specific user.  UserName is secret, so We use POST |
| 9 | **GetCarDetailsByID** | GET | *CarModelID (as parameter in the URL)* | {"CarSeriesID": *the new car’s series identifier*,  "Category": *the car’s category*,  "Manufacturer": *the car’s manufacturer*,  "ModelName": *the car’s model*,  "Year": *the car’s year of manufacture*,  "Color": *the car’s color*,  "Price": *the car’s price in NIS*,  "NumInStock": *number of available units*  } | The car’s details may take a lot of memory space (contains pictures etc.) so we don’t want to save it on the client side without reason. Using GET because there any wish to change any information |
| 10 | **GetCarsOfMenufucturer** | GET | *Manufacturer(as parameter in the URL*) | [  {  "CarSeriesID": *carSeriesID*  …  },  ] | The manufacturer are knows so we can use GET because there is no change in the DB |
| 11 | **getPrevOrders** | Post | {"username":<*user’s username*>} | [  {  "OrderID": *The orderID*  "Date": *the order’s date*  }…  ]  Or  {"False":" Not found any order"} | Show all previous Orders of users.  UserName is confidential , so We can't use GET and we'll use Post |
| 12 | **getCarsInOrder** | Post | {"OrderID":<*the order’s ID*>} | [  {  "CarSeriesID": "*the car’s series id*,  "Quantity": *number of units of this car in the order*  …  }  ]  or  {"False":" OrderId not exist"} | The orders store in the server because it the core of the system. The orderID is confidential , so We can't use GET and we'll use Post |
| 13 | **getOrderDetails** | Post | {"OrderID":<*the order’s ID*>} | {  "userName": *the username*,  "OrderID": *the orderid*,  "Date*": the order’s creation date*,  "Price": *the total price*,  "Currency": the currency>  "DeliveryDate<*the delivery date*}  OR  {"False":" OrderId not exist"} | The orders store in the server because it the core of the system. The orderID is confidential , so We can't use GET and we'll use Post |
| 14 | **CheckIsInStock** | POST | {"CarSeriesID":<*car’s series id*>  "Units":<*the wanted number of units*>} | {"True":" InStock"}  OR  {"False":" NotInStock"} | True = in stock, False = out of stock  POST because we don’t want that anyone would know the intention of the client to buy cars |
| 15 | **BuyCart** | POST | {  "Items": [  {  "<*carSeriesID*>":<*num of units*>,  "<*carSeriesID*>":<*num of units*>  }  ],  "username": <*the client’s userneme*>,  "totalPrice": <price>  "currency":<*currency*>,  "deliveryDate":<YYYY-MM-DD>  } | {"Sucess":*The new order number*}  Or  {"Failed":" *failure reason*} | \*user Buy the whole shopping cart  \*return error if the buying not succeeds.  return the orderID if succeed  \*the amount is transfer baucuse the curency Exchange will be done in the client side  POST because there is secret information to deliver to the server.  A new parches is saved in the user’s purchases in the DB.  \*the server check if the products in stock.  If not an error return to the client |
| 16 | **BuyOneCar** | POST | {  "carSeriesID":<carSeriesID>,  "username<username>,  "totalPrice": <the price>,  "currency":<*currency*>,  "deliveryDate":<YYYY-MM-DD>  } | {"Sucess":*The new order number*}  Or  {"Failed":*failure reason*} | User Can buy one car.  We use post because we don’t want that anyone know about the user’s purchases.  Currency can by USD, NIS etc.  The price will calculated in the client side according to the currency rate |
| 17 | **GetAllCategories** | GET | - | [  {  "Category": "Mini"  },  {  "Category": "Private"  },  {  "Category": "Sport"  },  {  "Category": "SUV"  },  {  "Category": "Truck"  }  ] | Get the cars’ categories.  There are no parameters to send so we can use GET |

הנחות

1. עגלת הקניות תישמר אצל הלקוח.
2. ה-cookie ישמר אצל הלקוח.
3. זיהוי המשתמש בשרת באמצעות ה-username.

**חלק בונוס – לא מומש בצד הקליינט רק בצד ה-server בעבודה 3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Method Name | HTTP Method | Parameters | Returns | Explanation |
| 18 | **AddAdmin** | PUT | {  “username”: <*user’s username*>  } | {"Sucess":"isAdmin"}  {"Failed":"username Not exist"} | The user’s record is updated so we use PUT |
| 19 | **addCar** | POST | {  "CarSeriesID": <*the new car’s series identifier*>  "cat": <*the car’s category*>,  "Manufacturer": <*the car’s manufacturer*>,  "ModelName": <*the car’s model*>,  "Year": <*the car’s year of manufacture*>,  "Color": <*the car’s color*>,  "Price": <*the car’s price in NIS*>,  "NumInStock": <*number of units*>  } | { "Sucess": "Car Added" }  OR  { "Failed": *failure reason*} | A new record is add to the DB so we’ll use POST |
| 21 | **deleteCar** | DELETE | {"CarSeriesID":<*the car SeriesID*>} | {"Success":"Deleted"}  Or  { "Failed": *failure reason*} | A record is Removed from the DB so we’ll use DELETE  There is no option to delete client with orders. |
| 21 | **deleteClient** | DELETE | {"username":<*the username to delete*>} | {"Success":"Deleted"}  Or  { "Failed": *failure reason*} | A record is Removed from the DB so we’ll use DELETE  True= success False=fail |
| 22 | **GetInventory** | GET | - | [  {  "CarSeriesID": *carSeries id*,  "Manufacturer": *cars’ manufacturer*,  "ModelName": *car’s modelName*,  "Year": *car’s year of manufacture*,  "NumInStock": *numbr of units in stock*  }, | The DB is not changed and no confidential parameter are sent to the server so use Get |
| 23 | **UpdateInventory** | PUT | {  "CarSeriesID": <*the car’s series identifier*>,  "NewUnitsToAdd":<*number of units to add*>  } | { "Sucess":"Stock was Updated" }  OR  { "Failed": <failed reason> } | The admin adds X new cars.  The Quantiry field of the specific car model is update in the DB so we’ll Use the PUT method  \*it can be possible to remove units by negative number of usnits |
| 24 | **GetAllOrders** | GET |  | [  {  "username": *the username*,  "OrderID": the *oreder’s number*,  "Date": the order’s date  "Price": *the order’s total price*,  "Currency": *the payment currency*,  "DeliveryDate": *the delivery’s date*  }…] | The admin can see all orders  For reports that will created in the client side |

הערות: לא הוספנו פונקציה חדשה להוספת קליינט כי היא זהה ל- register.

בכל מקום שיש שגיאה ב-DB היא תחזור בתור JSON. (לדוגמא מחיקת שדה שהינו foreign key)

מתכנת – נועה שוורץ

קשיים:

* קביעה איזה פונקציות יהיו בצד שרת ומה אצל הלקוח.
* באיזו מתודה של http להשתמש.

לקוח – לרה שטוטלנד

קשיים:

* להבהיר את כל הדרישות הקטנות של המערכת בשפה שתהיה ברורה למתכנתת.
* המרת חזון המערכת לדרישות.

תוספות ושינויים -עבודה4

1. הוספת הפונקציה GetAllCars.

-ב-API המקורי הפונקציה **GetListCarsOrderedByCategory** החזירה לנו רק את מזהה המכונית והקטגוריה.

נתקלנו בצורך לקבל את כל פרטי המכוניות בבת אחת על מנת לאתחל את מערך ה-products על מנת שיהיה לנו את המידע המלא על המכוניות בצד הקליינט ולא נצטרך לשלוף כמה פעמים את המידע מה-server. זאת על ידי Product Model שנוצר עבור כל מכונית.

1. החזרת Token בפונקציית ה-Login, ושליחת ה-Token בפונקציות GetRecommendsו- getPrevOrdersבשל הצורך לוודא את החיבור עם המשתמש בפונקציות בהן יש צורך בזיהוי המשתמש.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Explanation | Returns | Parameters | HTTP Method | Method Name | ID |
| Get all Cars Full Details | [  {  "CarSeriesID":… "Category":…  Manufacturer":…  ModelName":…  Year":…  Color": …  PriceNIS":…  NumInStock":…  DateOfEnter":…  },…  ] | - | GET | **GetAllCars** |  |
| Login cannot pass in GET | { "Success": *" token "* }  { "Failed": *failure reason*} | {"username":<*user’s username*>  "password: <*password*>} | POST | **Login** | **1** |
| Return recommends of car that fits to specific user.  Token and user name is confidential, so We use POST | [  {  "CarSeriesID": *carSeriesID*  …  },  Or  {"False":" *failure reason* "}  ] | {"username":<*user’s username*>,  "Token",< *user’s* token >} | POST | **GetRecommends** | **8** |
| Show all previous Orders of users.  Token and user name is confidential , so We can't use GET and we'll use Post | [  {  "OrderID": *The orderID*  "Date": *the order’s date*  }…  ]  Or  {"False":" *failure reason* "} | {"username":<*user’s username*>,  "Token",< *user’s* token >} | Post | **getPrevOrders** | 11 |