### Introduction

BAMRI ask you to make a minimum viable product for driver and bus scheduling system. Your task is to implement the backend with Laravel PHP Framework and interactive frontend with JavaScript to consume the backend API. The detail description and tools that you can use will be described below.

### Description of project and tasks

**API List:**

Use provided ERD to make your database. Create dummy users on users table (password is hashed using bcrypt):

|  |  |
| --- | --- |
| **username** | **password** |
| admin | passadmin |

These are the list of web service endpoint that requested by the company:

1. **Authentication**
   1. **Login**

URL: [domain]/v1/auth/login

Description: For admin to generate and get login token using username and password. Username and password must be valid.

Method: POST

Request Parameter:

* Body:
  + Username
  + Password

Response:

* If login success:

Header: response status: 200

Body: *{ token }* (authorization token generated by the system from logged in username with md5 encryption)

* If login failed (username or password do not match or empty):

Header: response status: 401

Body: *{ message: invalid login }*

* 1. **Logout**

URL: [domain]/v1/auth/logout?token={token}

Description: For server to make token invalid

Method: GET

Response:

* If logout success:

Header: response status: 200

Body: *{ message: logout success }*

* If logout failed (token invalid):

Header: response status: 401

Body: *{ message: unauthorized user }*

1. **Bus**
   1. **Create new bus**

URL: [domain]/v1/buses?token={token}

Description: For admin to create new bus.

Method: POST

Request Parameter:

* Body:
  + Plate Number *(string; required)*
  + Brand *(mercedes / fuso / scania; required)*
  + Seat *(integer; min:1; required)*
  + Price Per Day *(integer; min:100000; required)*

Response:

* If success:

Header: response status: 200

Body: *{ message: create bus success }*

* If input validation failed:

Header: response status: 422

Body: *{ message: invalid field }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Update bus**

URL: [domain]/v1/buses/{*bus\_id*}?token={token}

Description: For admin to update existing bus by id.

Method: PUT

Request Parameter:

* Body:
  + Plate Number *(string)*
  + Brand *(mercedes / fuso / scania)*
  + Seat *(integer; min:1)*
  + Price Per Day *(integer; min:100000)*

Response:

* If success:

Header: response status: 200

Body: *{ message: update bus success }*

* If input validation failed:

Header: response status: 422

Body: *{ message: invalid field }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Delete bus**

URL: [domain]/v1/buses/{*bus\_id*}?token={token}

Description: For admin to delete existing branch by id

Method: DELETE

Response:

* If success:

Header: response status: 200

Body: *{ message: delete bus success }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Get all bus**

URL: [domain]/v1/buses?token={token}

Description: For admin to get all bus data

Method: GET

Response:

* If success:

Header: response status: 200

Body: all bus data in json *{ id, plate\_number, brand, seat, price\_per\_day }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

1. **Driver**
   1. **Create new driver**

URL: [domain]/v1/drivers?token={token}

Description: For admin to create new driver.

Method: POST

Request Parameter:

* Body:
  + Name *(string; required)*
  + Age *(integer; min:18; required)*
  + ID Number *(string; 16 character; required)*

Response:

* If success:

Header: response status: 200

Body: *{ message: create driver success }*

* If input validation failed:

Header: response status: 422

Body: *{ message: invalid field }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Update driver**

URL: [domain]/v1/drivers/{*driver\_id*}?token={token}

Description: For admin to update existing studio by id.

Method: PUT

Request Parameter:

* Body:
  + Name *(string)*
  + Age *(integer; min:18)*
  + ID Number *(string; 16 character)*

Response:

* If success:

Header: response status: 200

Body: *{ message: update driver success }*

* If input validation failed:

Header: response status: 422

Body: *{ message: invalid field }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Delete driver**

URL: [domain]/v1/drivers/{*driver\_id*}?token={token}

Description: For admin to delete existing driver by id

Method: DELETE

Response:

* If success:

Header: response status: 200

Body: *{ message: delete driver success }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Get all driver**

URL: [domain]/v1/drivers?token={token}

Description: For admin to get all drivers data

Method: GET

Response:

* If success:

Header: response status: 200

Body: all driver data in json *{ id, name, age, id\_number }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

1. **Order**
   1. **Create new order**

URL: [domain]/v1/orders?token={token}

Description: For admin to create new order. For each order, match available bus with available driver in the given date. Make sure that there is no conflicting order (one bus in 2 different dates). For example, if bus A and driver D is rented from 1 January 2018 for 5 days, both bus A and driver D cannot be ordered anymore until 5 January 2018 because it already rented from 1 January until 5 January.

Method: POST

Request Parameter:

* Body:
  + Contact Name *(string; required)*
  + Contact Phone *(string; digits; required)*
  + Start Rent Date *(date; min:tomorrow; required)*
  + Total Rent Days *(integer; min:1; required)*

Response:

* If success:

Header: response status: 200

Body: order data in json with relationship of ***bus*** and ***driver*** *{ id, contact\_name, contact\_phone, start\_rent\_date, total\_rent\_days, bus\_id, driver\_id, bus, driver }*

* If input validation failed:

Header: response status: 422

Body: *{ message: invalid field }*

* If order cannot be fulfilled because conflicting order or no available bus or driver:

Header: response status: 401

Body: *{ message: conflicting order or no available bus or driver }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Delete order**

URL: [domain]/v1/orders/*{order\_id}*?token={token}

Description: For admin to delete existing order by id

Method: DELETE

Response:

* If success:

Header: response status: 200

Body: *{ message: delete order success }*

* If accessed without token:

Header: response status: 403

Body: *{ message: forbidden }*

* 1. **Get all orders**

URL: [domain]/v1/order?token={token}

Description: For admin to get all orders data

Method: GET

Response:

* If success:

Header: response status: 200

Body: all order data in json with relationship of ***bus*** and ***driver*** *{ id, contact\_name, contact\_phone, start\_rent\_date, total\_rent\_days, bus\_id, driver\_id, bus, driver }*

* If accessed without token:

Header: response status: 403

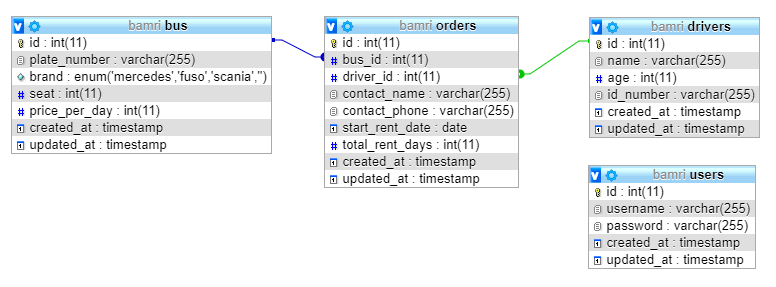
Body: *{ message: forbidden }*

The complete BAMRI system should cover the following requirement:

|  |  |
| --- | --- |
| **Menu** | **Detail** |
| Login | * The user can login (and logout) into the system on the start page of the application * After login, user have access to “bus”, “driver”, and “order” menu * Links to the authorized menu and logout link are always visible on the top part of the page while the user is logged in * Make sure your system is preventing users to access unauthorized menu * There must be no register feature available |
| Bus | * On “bus” menu, user can add new bus * User can update and delete existing bus * All input form is must be **filled** on **create** * **Plate** **Number** must be a *string* * **Brand** must be between *mercedes / fuso / scania* * **Seat** must be *integer* and *more or equals 1* * **Price per day** must be *integer* and *more or equals 100.000* |
| Driver | * On “driver” menu, user can add new driver * User can update and delete existing driver * All input form must be **filled** on **create** * **Name** must be a *string* * **Age** must be *integer* and *more or equals 18* * **ID Number** must be a *string* and *consists of 16 character* |
| Order | * On “order” menu, user can add new order * User can update and delete existing order * All input form is must be **filled** on **create** * **Contact Name** must be *string* * **Contact Phone** must be *string* and *digits* * **Start Rent Date** must be *date* and *minimum date is tomorrow* * **Total Rent Days** must be *integer* and *more or equals 1* * The system will match available bus with available driver. * No bus and driver rent time can’t be **overlap** with other order. |

**ERD**

You can use and improve ERD below:



### Instructions to the Competitor

* Save your files as "**XX\_MODULE\_SERVER.zip**" and upload to the server where XX is your registration number.
* You don’t have to put vendor and node\_modules folders on your zip file.

## Create/generate a DB-diagram named “db-diagram.xxx” (xxx is the extension/type of the file eg. pdf or jpg) and put it into the directory mentioned above. Example:

## For this module, you must use one of the three available frameworks provided. Applications developed without use of any of these frameworks will not be considered. You should take advantage of the framework as much as possible.

## Assessment will be done on the files and data in your database “XX\_MODULE\_SERVER"