**TRANSPORT MANAGEMENT SYSTEM**

A company wants to automate the vehicle allocation system. Every employee can be allocated to a vehicle that has seats open.

The system provided the below capabilities,

1. **Setup Admin Information (**User Name, Password**)**
2. **Setup Employee Information** (Employee ID, Name, Age, Location, Phone, Vehicle ID)
3. **Setup Vehicle Information** (Vehicle Number, Capacity, Available seats, Is Operable)
4. **Setup Route Information (**Root Number, Vehicle Number, Stops**)**
5. Allocate a vehicle to employee

## **Detailed Requirements**

Admin Setup

We need to create a database table to store the admin data like admin User\_Name and password which will fetch/match the admin information while logging in the application, the data table will include the following filed:

1. User name
2. Password

Display 1: Login

Once the application page is clicked by the admin, it will take him to login-page (with logo) and prompt him to enter his username and password. The admin user name and password will be validated at the server side against the database table – if the details match, the admin will be redirected to the Home page and if not, he will get the same login page with the error message “You’ve entered an incorrect user name or password”.

Display 2: Home Page

This home page will be shown only when the admin has successfully logged in. This will have the application logo/image and also a navigation bar which will have the below options,

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **ACTION NAME** | **ACTION ON** |
| 1 | Add | Vehicle, Employee, Add Routes |
| 2 | Modify | Vehicle, Employee |
| 3 | Allocate Vehicle | Allocate Vehicle to Employee |
| 4 | Change Vehicle | Change the status of the vehicle |
| 5 | Logout | Logout from the application |

Add Employee

Each employee will have a unique Employee Id and Name, so to manage the employee data we need to create a database, so it can manage the data directly. The Employee database table will have the following attributes which will contain the particular field. So once the admin will click on “Add Employee”, he will need the following fields:

1. Employee ID
2. Employee Name
3. Age
4. Location
5. Phone
6. Vehicle ID

Add Vehicle

Admin will use this screen to add Vehicle information to the system. Screen will have the below fields,

1. Vehicle Number (Unique)
2. Capacity (how many employees/passengers can occupy the seat)
3. Available seats (Is any seat available?)
4. Operable (Is this vehicle operable from this to that location)

Add Route/Change Route

Admin will use this screen to add Route information to the system. Screen will have the below

fields,

1. Root Number (Unique)
2. Vehicle Number
3. Stop-1, Stop-2, ……..

Modify:

Similarly, the admin can modify the employee, vehicle and route based on requirements

Allocate vehicle:

Once the all fields are stored admin will allocate the vehicle for the employee

### General Requirements

* When an operation is successful, the admin should be shown a message as such. E.g., if the “Add Vehicle” is successful, it will display “Vehicle successfully added to the system”.
* On failures, the admin should be shown appropriate friendly error message.

### Technical Requirements

* Tables should be designed appropriately (proper data types, table/column names, constraints), use a SQL Server database
* The tables should be created only using SQL Files and these should be available in the repo
* Front end – ASP.NET Core MVC, HTML5, CSS, Bootstrap, Javascript/Jquery
* Business Logic – Web API/Service in MVC
* Data Access – EF Core (DB/Model first approach)
* All names used in the code (classes, variables, properties etc.,) should be meaningful and follow consistent naming approach (Camel Case, Pascal Case etc.,)
* Classes/methods/properties should have proper accessibility modifiers
* Use code formatting to ensure the code is readable, write comments where required so that the reviewers can understand your code

### IMPORTANT

* Create a new repo “ProjectA” in your account and use that for this exercise
* Push the code to the remote repository often or at least before end of each day.
* Repo should have a folder “DB Script” and it will have all the SQL Files (tables, stored procedure etc.,)
* Repo should have a folder “src” and it will have your ASP.NET Core Projects