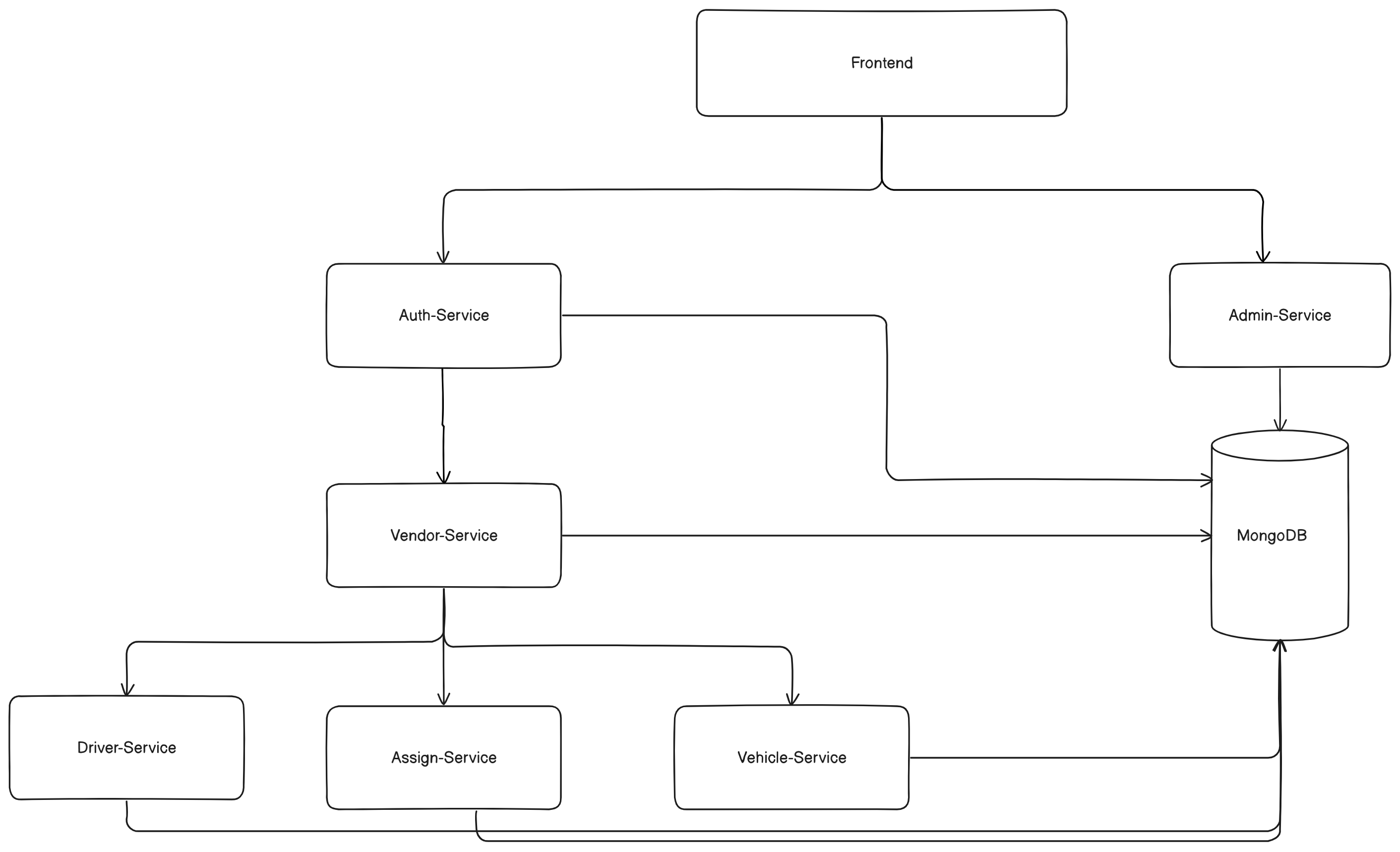
**Vendor Cab and Driver Onboarding & Vendor**

**Hierarchy Management**

Architecture:

  
1. Auth-Service: Manages authentication. The authentication is based on JWT (Json web token).

2. Vendor-Service: Manages vendor hierarchy and vendor creation.

3. Driver-Service: Manges driver creation, deletion, update.

4. Assign-Service: Helps in assigning vehicles to drivers.

5. Vehicle-Service: Manages vehicles.

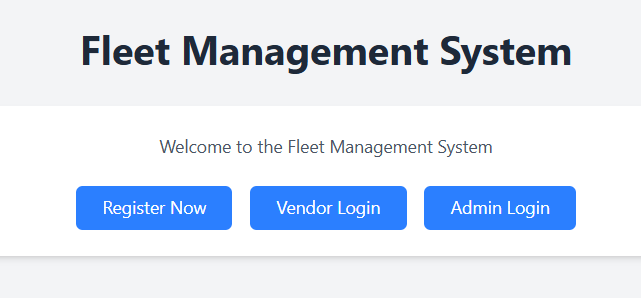
6. Admin-Service: All admins are managed by this service.

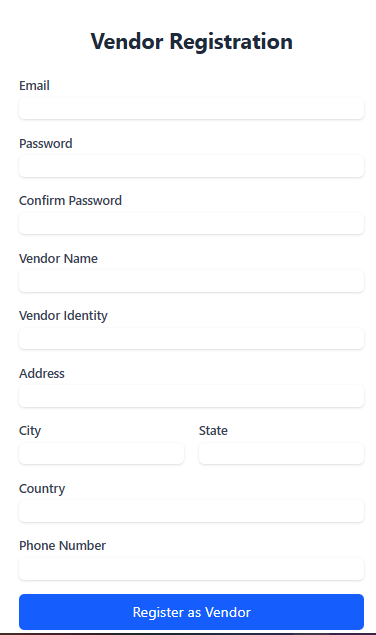
Microservices Architecture-

1. Each service (Auth-Service, Admin-Service, Vendor-Service, Driver-Service, Vehicle-Service, Assign-Service) is a separate and independent component.
2. All services are loosely coupled so that if one fails other works. (Example: Assume the driver service failed still our admin can perform actions on drivers as it is not dependent on driver service.)
3. Frontend can communicate with different services but it needs to go through authentication every time.
4. While there's a logical hierarchy (e.g., Vendor-Service spawning Driver/Vehicle/Assign services). Services can still communicate directly with the database. No tight dependencies between services
5. Services can be distributed across different servers. Each service can be independently scaled. No direct service-to-service code dependencies.

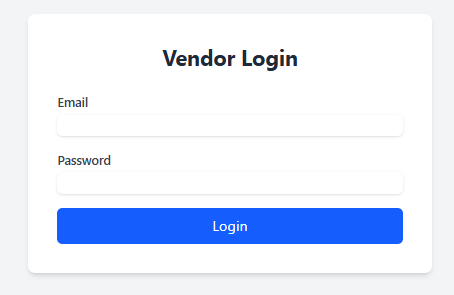
Application flow-

Frontend

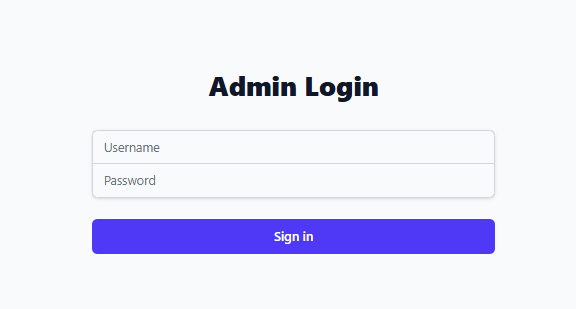


Register Now takes you to vendor registration  
 

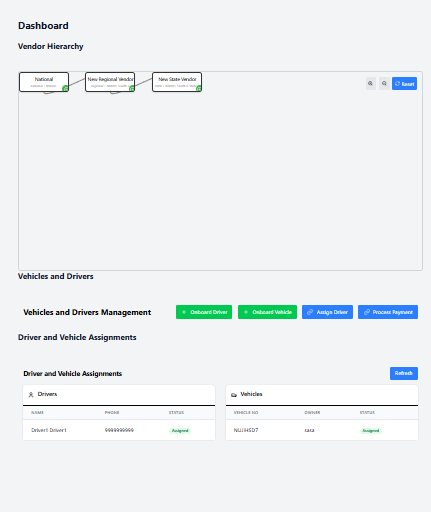
Vendor Login takes you to vendor login



Admin Login



After successful registration and login

  
All vendor can create mores sub vendor below their hierarchy.

Vehicle and driver management

Hierarchy Management



Whenever vendor is registered it is registered as a national vendor, he can create any number of regional vendors and give permission to them for simplicity there are 5 permissions.   
1. Create sub vendor

2. Onboard Driver

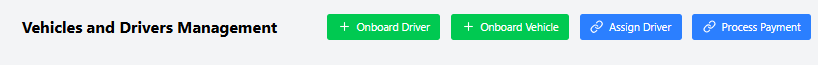
3. Onboard Vehicle

4. Assign Vehicles

* This is vendor dashboard:

1. Hierarchy management- allows vendor to visualize the hierarchy of vendors below it. He can grant and revoke permissions to the sub-vendor below it.
2. Permission granted can only be passed below. (Example if regional vendor has permission to process payments than only, he can pass it to state vendor).
3. If the permission is revoked from a vendor our system automatically revokes that permission from sub-vendors below its hierarchy.
4. Vendor can directly add sub-vendors below its hierarchy by the green plus button. The vendor will create passwords for sub vendors to ensure safety.
5. Vendor can only delete sub-vendors if they do not have any active vendor below their hierarchy.

* Vehicle and driver management:

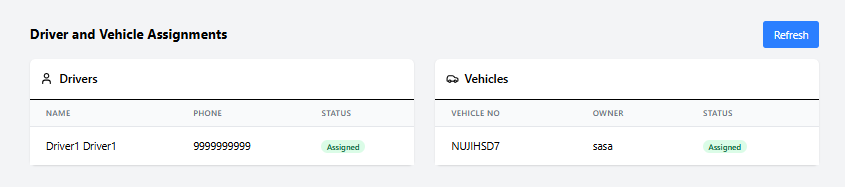


In this section if vendor has permission user can

1. Onboard Driver
2. Onboard Vehicle
3. Assign Driver
4. Process Payment (it is only a button)

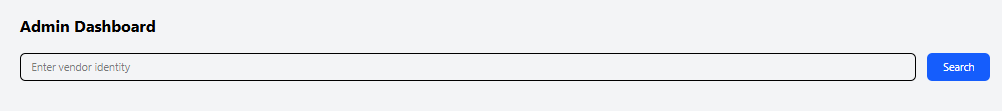
The all 4 buttons block or on block on the basis of permission the vendor has

* Driver and vehicle assignments:



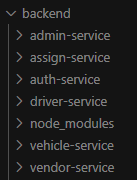
1. In this section user can easily visualize which driver and vehicle is active.
2. Which vehicle is assigned which vehicle is yet to be assigned.
3. Which driver is free and can be assigned.

* Admin Dashboard:



1. The search bar is for searching the unique id which is made by the main head of vendors during registration.
2. By searching the id of the vendor, admin can view all the vendors of that identity and all vehicles onboarded under that identity.

Backend-



Auth service:

1. Logs: Contains logs
2. Controllers
   1. auth.controller.js- contains controller register, login, getme.
3. Middleware
   1. Auth middleware- protect (check req cookies for token if token found validate it and add vendor data to request).
   2. errorHandler- Extends error class as AppError and handles all major error.
4. Models
   1. User model contains email, password, role, vendorId (referenced to vendor schema), isActive, lastLogin.
5. Routes
   1. /login
   2. /register
   3. /me (it is protected by middleware)
6. Index.js main entry point

Auth-service – allow vendor for authentication any vendor register is registered as a head vendor who than create multiple sub-vendors and so on.

Vendor-Service:

1. vendor.controller : verifyVendor, loginVendor, logoutVendor, createVendor, registerVendor, getVendors, getVendorHierarchyTree, updateVendorPermissions, registerSubVendor, registerNationalVendor, grantPermission, revokePermission
2. auth: protect (check req cookies for token if token found validate it and add vendor data to request).
3. vendor.model : name, email, password, vendorLevel, identity, parentId, isActive, permissions.createVendors, permissions.processPayments, permissions.vehicleOnboarding, permissions.driverOnboarding, permissions.assignVehicle, lastLogin, deletedAt, location.address, location.city, location.state, location.country, contact.phone, contact.alternatePhone, hierarchyLevel, grantedPermissions
4. vendor.routes:

1. Authentication Routes

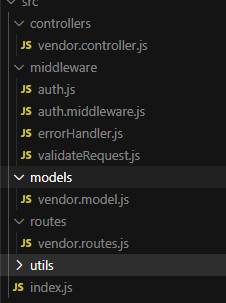
2. Vendor Registration Routes

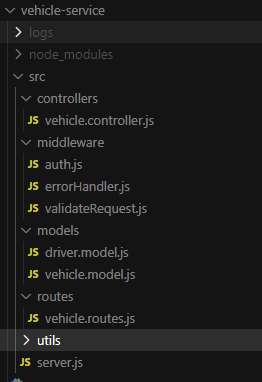
3. Vendor Management Routes

4. Permission Management Routes

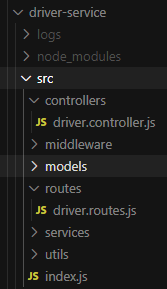
5. Hierarchy Routes

Protected Routes (require auth middleware):

All routes except: - /login - /register/national - /verify

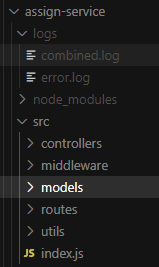
Vehicle-Service

1. vehicle.controller
2. auth.js
3. vehicle.model
4. vehicle.route
5. server.js

Driver-Service

1. driver.controller
2. Middlewares
3. Models
4. driver.routes
5. index.js

Assign-Service

1. controllers
2. Middlewares
3. Models
4. Routes
5. Utils
6. Index.js