Data & Entity Modelling Design Diagrams

The prototype VMS application contains two entities namely Vehicle and Driver. Each entity has fields which correspond to data e.g vehicle's speed. Vehicle and Driver entity forms a has-a relationship. Every driver has a vehicle.

The following figure shows the ER model describes the structure of the database.

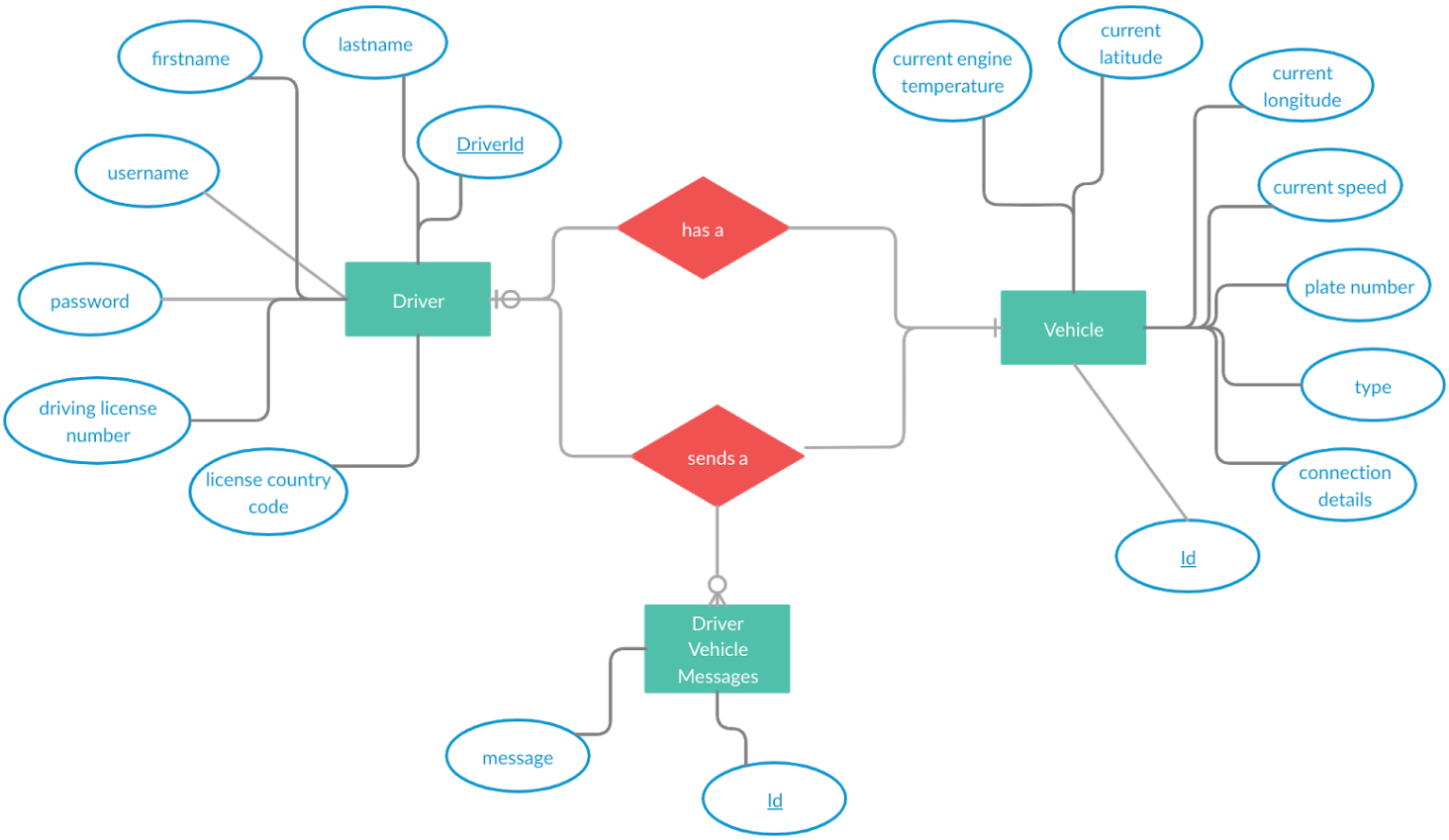


Fig: ER Model

The Below Physical data model shows the schema, relationships between tables, primary key and foreign key.

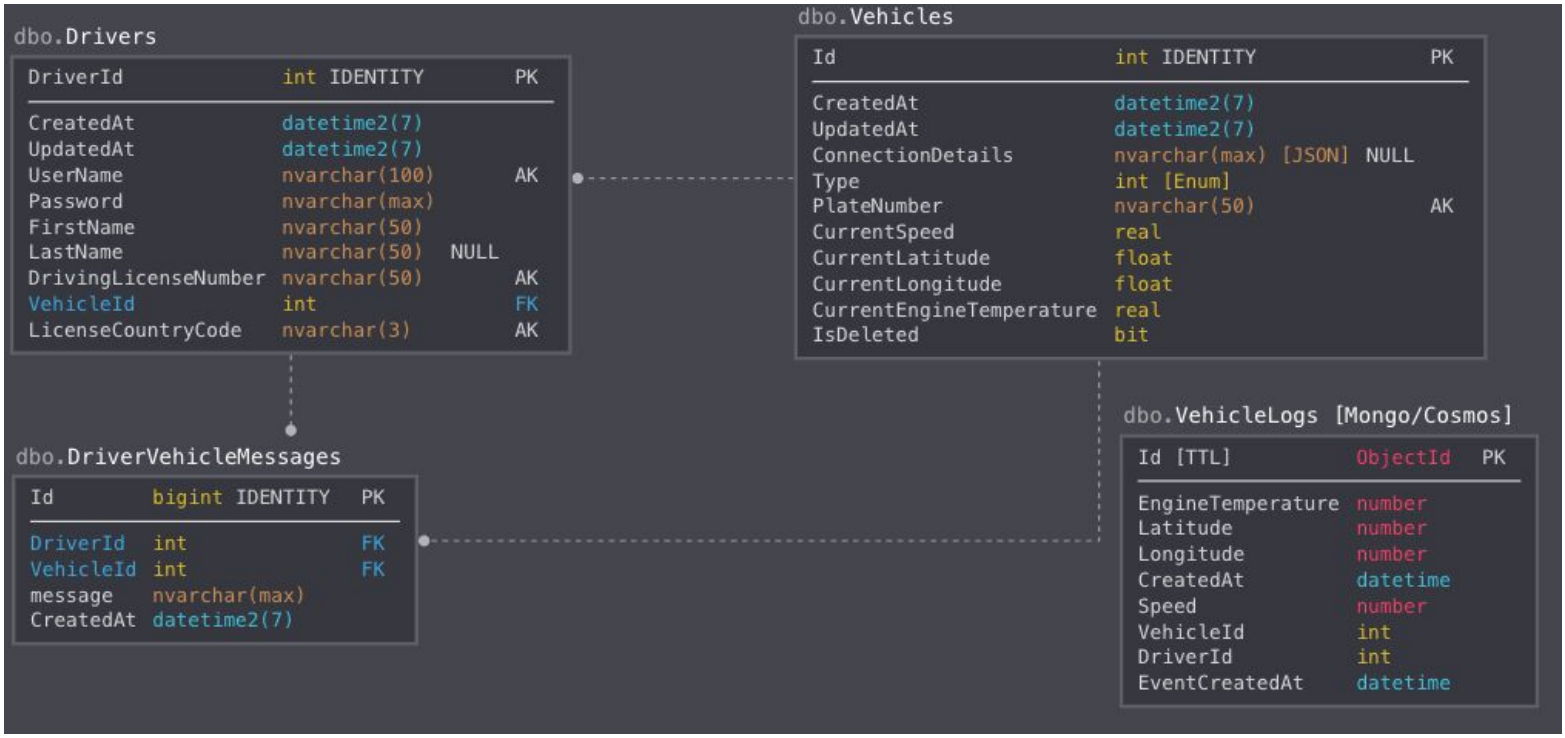


Fig. Physical Data Model

Driver Table: stores the information related to the vehicle driver.
Vehicle Table: stores the information related to the vehicle and current state of the vehicle.
Driver Vehicle Messages Table: stores the communication messages received by the driver and/or the vehicle owned by the driver in that duration.
Vehicle Logs Table: A no-sql collection that stores high-bandwidth vehicle logs. Benefits being ability to handle large volumes of rapidly changing structured, semi-structured, and unstructured data.

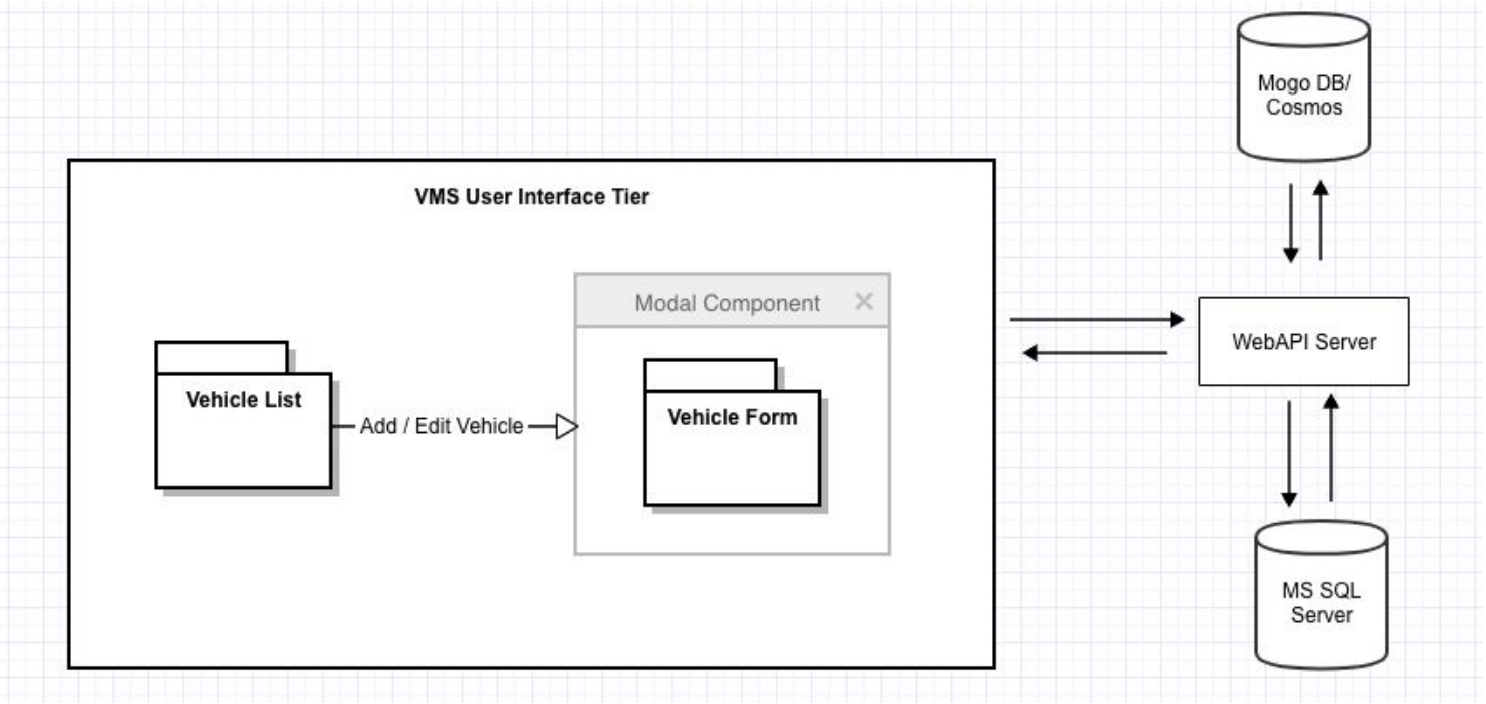
System Design & Flow

The application follows a 3-tier architecture.

1. User Interface tier (Web) - React App
2. Service tier (Backend) - Asp.Net Core WebAPI
3. Database tier - Microsoft SQL Server

The react app comprises of 3 main components

1. Vehicle List component - This shows the list of vehicles currently available in the VMS.
2. Vehicle Form component - This component handles creating & updating functionalities inside a Modal component using React portal.
3. Redux - It manages the state across react components.



For the database tier SQL Server and MongoDB

MongoDB database is being used to save storing Vehicle sensor logs. Opting for a NoSQL DB is advantageous in this case as it can flexibly handle frequent schema changes and high velocity of transactions. Also, with growing business needs, MongoDB can easily be distributed across geographical regions providing a better and faster availability while maintaining reliability. This will also help in rich analytics and data mining.

For the rest of the application flow RDBMS is an optimal choice, as for this part the data needs to be consistent. Main areas where SQL Server will shine is storing complex entity relationships while ensuring ACID compliance and features like joins, stored procedures and triggers make querying these entities a breeze.

Assumptions

1. All vehicle data/information is collected through a central unit, i.e, sensors do not directly transfer data to the server, instead a centralized unit [Sensor Hub] combines data from all the sensors (pressure, fuel etc) as a single transactional unit to the endpoint.
2. Basic necessary information (location, speed) is uploaded by every sensor hub.
3. Vehicle sensor logs are stored in No-SQL database for faster and scalable analysis, data storing and data insights. Logs are rotated periodically depending on business requirements.
4. Time Zone information is not recorded. All timezone is assumed to be in UTC.
5. Vehicle types are static enum values (Truck, Van) as of now.