**Vendor Module Overview**

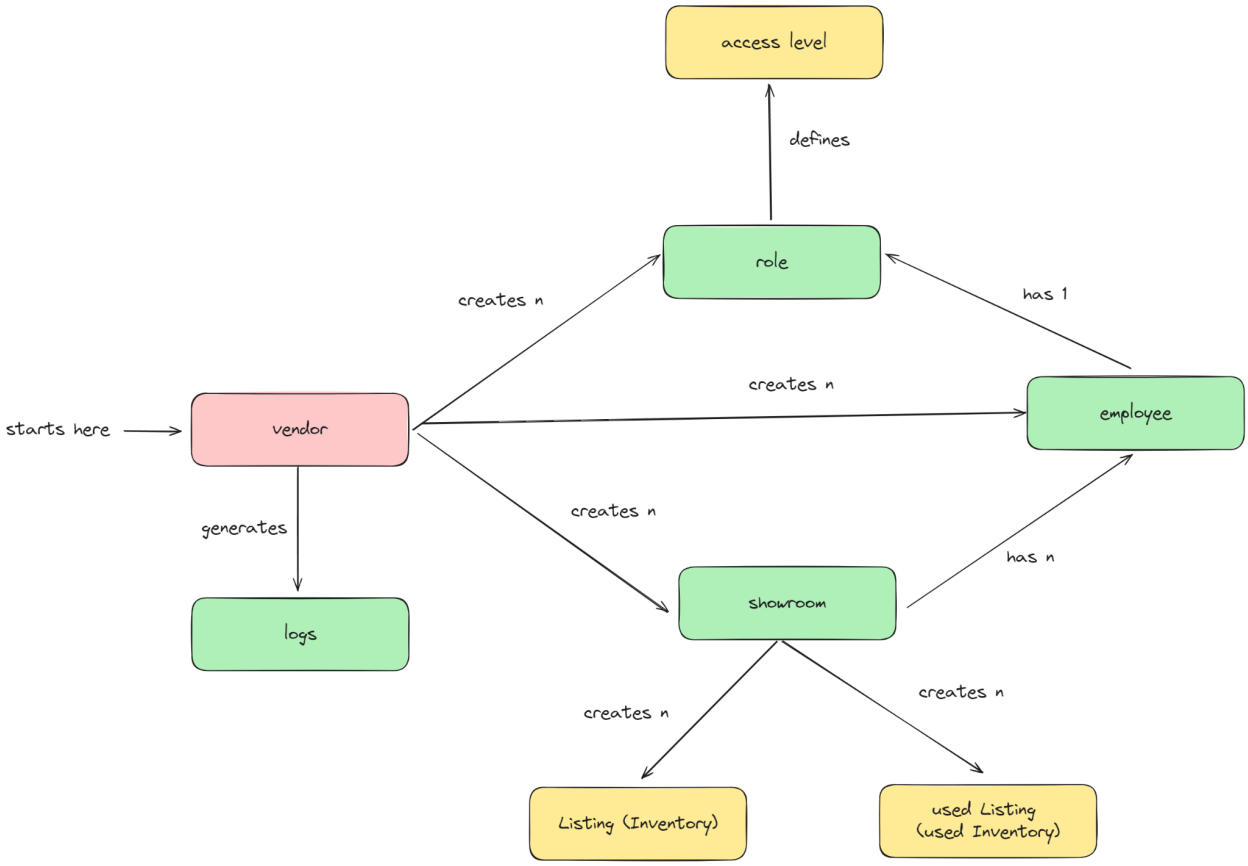



Fig: Vendor Module Architecture

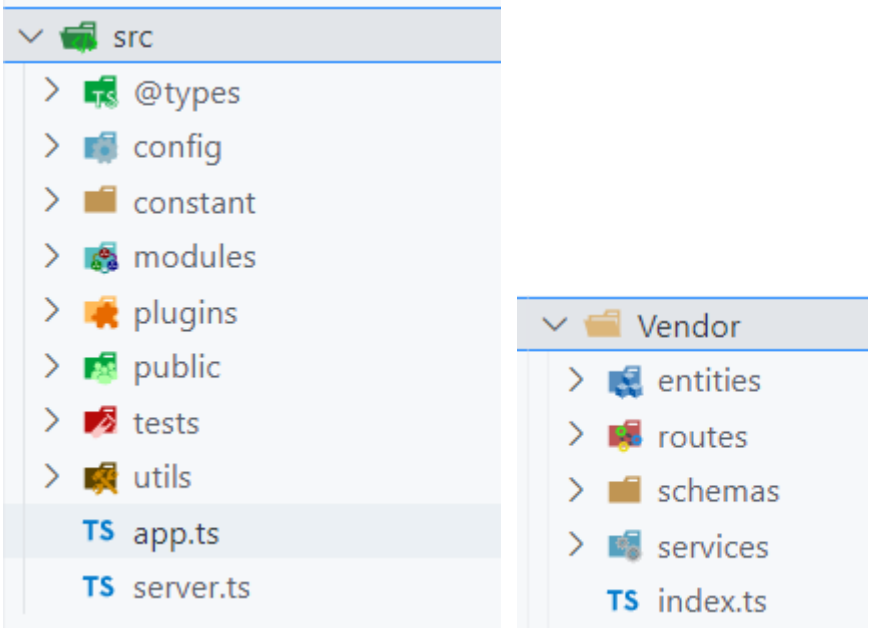
Overall it explains how this vendor module is designed. Each entity and its relations are explained in the above diagram.
Each vendor has many roles, employees and showroom. Each action performed will register a log. Each showroom has its own set of employees, new and old listings. Each employee has a role assigned to them.

**Authentication & Authorization:**

For authentication: Email and Password is used for authentication and one verification step is used by sending tokens to the user `email`. It's a 2FA strategy for authentication.

For authorization: JWT using cookies is used for authorization strategy.

**Project Structure & Module Structure:**




**# Project Structure Explained:**
This explains all the directory in `src` used to separate the files related to the project.

**@types**
Its contains all the type required for the project.

**config**
It contains all the configuration for the database and other required config for the project.

**constants**
It contains all the constant data for the project.

**modules**
It contains all the modules/domains required for the project. Each module is a mini project of its own and the api routes are generated based on the module.

**plugins**
It contains all the plugins/add-on functions or logic added to the **fastify** framework.

**test**
It contains all the test files.

**utils**

It contains all the utility files required for the project from aws, string, data and so on utility functions.

**# Module Explanation:**
It's an explanation of a module. Each module has its own routing, entities/modals, schemas, service/logic and a start file `index.ts`.

**entities**
It contains all the db models/entities that are required for the model. It's a format of what the database will look like. We are using `typeorm` for this with SQL database.

**routes**
It contains all the routes/api endpoints required for the module.

**schemas**
It contains all the schema used to validate the user's input/request.

**services**
It contains all the logic required for the manipulation of data and request handlers.

**index.ts** - each module has their main entry point to manage the module.