Back-End Infrastructure Implementation

This lesson explains the back-end infrastructure implementation, the model's folder, the working of mongoose, and the user.js file.

WE'LL COVER THE FOLLOWING

In the previous lesson, we discussed the frontend infrastructure. Now, let's look at the back-end.

model Folder

In this project, you will find the directory *mean_backend*. If you look into it, you'll see that there is a *user model* defined on the back-end as well (`/model/user.js` file).

Here, is where the library **Mongoose** comes into play. This library is used to simplify validations, castings, and business logic that revolves around database access to MongoDB. There is a folder *model* on the back-end implementation too, but this one is a bit more complicated since it contains a database access layer as well.

This is what it looks like:

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

// Define a schema.
var userSchema = new Schema({
  name: String,
  blog: String,
  age: Number,
  location: String
});
```

```
// Define a method for concatanation of name and blog fields.
userSchema.methods.concatanceNameAndBlog = function() {
    // Extend name with value of the blog field.

    this.name = this.name + this.blog;

    return this.name;
  };

// Create a model.
var User = mongoose.model('User', userSchema);

module.exports = User;
```

/mean_backend/model/user.js

Explanation

In the file above, we used *Mongoose* to define *user schema* (**line 5**). One of the cool things with about this is that you can define functions in your schema as well, as shown with the example of `concatanceNameAndBlog` function (**line 13**).

Also, Mongoose gives various other possibilities for defining *required fields*, *read-only fields*, *virtual fields* and *indexes* as well.

You can check it out on their website.

After the schema is defined, we created a `User` model (**line 21**), which was then exported (**line 23**). As you will see in the next lessons, we will do out all of the queries in the database with this model.

Also, note that the *routes* folder is inside the *mean_backend* folder, which contains the *index.js* file inside of it. This file will contain our Web API, which will be called from the front-end.

Now that you know about the front-end as well as the back-end infrastructure, let's move to the CRUD operations. In the next lesson, you will learn how to create a user.