Tutorial 09: MongoDB and Mongoose

NOTE: You may refer to sample codes shared on lecture notes for more details to create a new express project.

Task: Post Request, MongoDB and Mongoose

1. Create a new project directory named "**tutorial9mongodb**" and change to the new directory.

```
mkdir tutorial9mongodb
cd tutorial9mongodb
```

2. Inside your project folder create two sub-directories named:
   a. backend
   b. frontend

3. Move to backend directory: cd backend
4. Inside the backend directory:
   a. Create a new package.json file (**-y** will set the default value for all information)

```
npm init -y
```

   b. Create a new main app.js file:

```
fsutil file createnew index.js 0
```

(**NOTE:** `index.js` file contains the gateway to your project, it contains the global project and express settings, it also has the code to connect to your database.)

5. Install all packages needed by backend:
   a. express
   b. mongodb
   c. mongoose
   d. nodemon (automatically restarts the node application when file changes in the directory are detected)

```
npm install express mongodb mongoose nodemon
```

6. Edit **package.json** file to:
   a. create a script for **nodemon** command to restart server automatically when if there are any changes in our script

```
"scripts": {
   "server": "nodemon server.js"
 },
```

b. To **start the nodemon**, execute this command at terminal:
```
npm run server
```

7. Edit index.js file to tell Express to:
   a. Require express and define a variable app to execute express as a function
   b. Define app.listen to start the server at a particular port
   c. Require “body-parser” and use body-parser (help us to read POST data)
   d. Define a root route

```
const express = require('express');
var bodyParser = require('body-parser');

const app = express();
const port = 3000;        //define a variable port to save
the port number

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

//Root route
app.get('/', (req, res) => {
  res.send('Hello World, Express.js with MongoDB!!')
})

app.listen(port, () => {
    console.log(`App has started on port ${port}!`)
  })
```

8. Setup Mongodb database
   a. Create a new database named ‘tutorial9_db’
   b. Create a new collection in the database named ‘transactions’

9. Using Mongoose to define schema and create model
   a. **Edit main server.js file to:**
      i. Add connection to the MongoDB (Example of connection link for a local server: 'mongodb://localhost:27017/tutorial9_db').
      ii. Define a new Mongoose schema for the Transaction which has five fields (i.e. date, type, description, amount, status). The date is the date when the users enter a new expense transaction.
      iii. Create a model for the Transaction using the schema defined in (ii).

10. Implement routes for creating and reading expense transaction records (Post and Get methods).
    a. Define a POST route called “/transaction/” to response by adding a new transaction document from Mongodb database

b. Define a GET route called “/transaction” to response by rendering all the transaction documents from Mongodb database

11. Start your web server and establish connection to mongodb using `nodemon` by executing this command at terminal to run the script:

```
npm run server
```