# ExpressJS Exercises: Introduction to ExpressJS

Refactor the 'MemeDB' project that is provided, so that your app meets the following requirements:

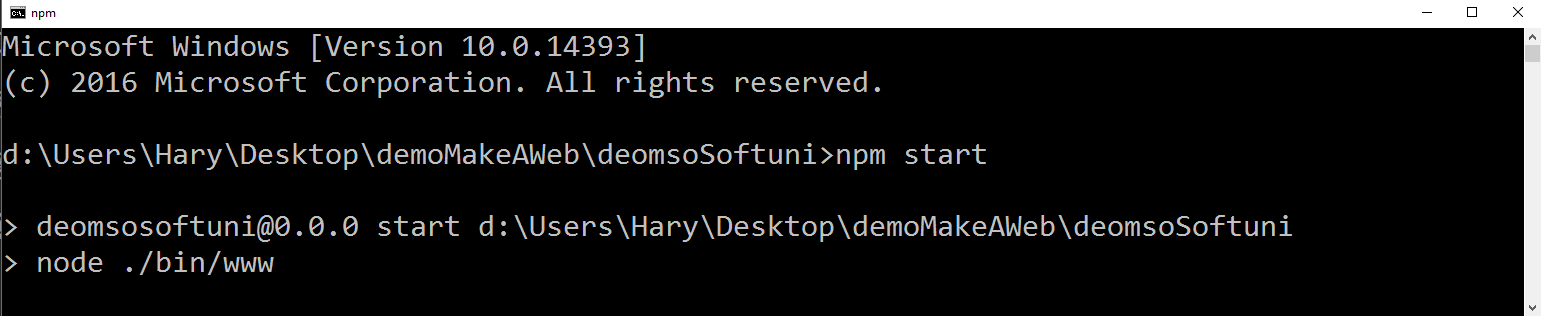
**Minimum requirements:**

* Have at least **6** **pages**
* Use **MongoDB**
* Have at least **2 Schemas**
* Use **Express**
* Use **File Upload**
* *(Optional)* Add a **REST API** for interacting with the database

You can start from scratch or use the provided project scaffold.

## Refactor the app

#### Configure MongoDB

Create **config** folder, inside of it create **dbConfig.js** module, which should be holding all your MongoDB settings. Connect the module at the appropriate place, so that it establishes connection when the app is started, also try to log a friendly message if the connection is successful.

#### Implement Routing

Create routes, that lead to the appropriate handlers, when a request is send.

#### Create Handlers

For each route, create a handler, that executes the desired back-end logic, when called.

\***Before** implementing any **upload** logic create all needed **Schemas**

## Create all needed Schemas

#### 3.1 Create genre Schema

Create genre schema that will hold genreName**,** array that holds all memes with the current genre memeList**.**

#### 3.2 Create meme Schema

Create meme Schema, that will hold **memeName, dateOfCreation, votes** which will hold all generated votes**, description, title.**

## Implement creation logic

#### 4.1 Implement logic for creating genre

Implement handler that, will save all send data in the DB, upon creation of a new genre.

#### 4.2 Implement upload logic

Implement upload logic, that will save all send data (it should be a **real** **image**, not a URL), upon creation you should fill the new entry’s id in the appropriate genre.

\* You must restrict the creation of new memes in such way, that the user can only create memes, with valid genre properties (only if the genre exists)

## Add REST API

Create a modular router that is mounted at /api that allows an application to perform CRUD operations with the database. Test the endpoints with Postman. Additionally, you may create a front-end script that fetches data from the API, instead of refreshing the page.