Integrating MongoDB with a Node and Express app

We can either install mongodb driver (from MongoDB team) or mongoose (third-party) for his integration. Mongoose has advantages like schemas and models (for imposing schemas on collections), validation, ability to hook into various operations (do something, say, while saving a document) etc.

Please refer https://docs.mongodb.com/drivers/node/quick-start to get started using mongodb driver.

We shall however use mongoose - https://mongoosejs.com/docs/index.html - Please check this page for further details of the steps discussed below.

Steps (at a high-level)

1. Install mongoose
2. Make sure MongoDB is running (mongod - the server)
3. Connect to the DB using mongoose - we can write a data/init.js script within which we can connect. The mongoose.connection object is an event emitter that emits appropriate events when it tries to connect. Ideally you will have to exit the application (process.exit()) on failure to connect, and also handle disconection as per your needs (we have not covered this).
4. For each of your collections in the DB, create a model in models/ folder. First define a schema object (instance of mongoose.Schema). Then use it to define a model using mongoose.model - the model is registered on the mongoose object with the name you supply here.
5. Make sure the DB setup (say setup in a file like data/init.js), and the Mongoose model files (say defined in a folder like models/*) are executed before any other scripts during application startup. You may include the model files within init.js and init.js at the top of index.js (the startup file for the Node app) for this.
6. Handle routes for various CRUD operations on the resource associated with a model - for example GET, POST, PATCH/PUT, DELETE on /workshops (or /workshops/:id).
    a. Make sure errors in request / internal error during DB operations etc. are handled well. For this you may pass the error object if available, or create one and pass it to the error handler middleware using next(error_object). Make sure you return from the function handling the request in such cases
    b. If all good, then return the result in response
    c. Always use appropriate error / success HTTP codes
7. Add the router to the app in Node startup file - index.js. You can mount it on a path (eg. /api/workshops if /workshops has not been prefixed for the routes, or /api if they have been prefixed for the routes) - all routes within the router have this path prefixed to the routes they handle.