# Lab2 - Express server

## 1- Provide REST API implementing CRUD operations on TODOs

- Use a filesystem, same as lab1.
- You must use middleware to validate inputs.
- You must return descriptive status codes in case of errors.
    - Examples
        - GET /todos
        - GET /todos/1
        - POST /todos
        - DELETE /todos/1
        - PATCH /todos/1

## 2- Provide a Static files server

- Download this image and serve it using express static.

## Bonus:

1- Add status to todo with default value ‘to-do’
2- Can change status to ‘in progress’ and ‘done’
3- Can filter todos using  queryparam /todos?status=”

## Notes:

- **Make sure to try the debugging environment**
- **Make sure that you enable restart on change using nodemon**
- **Make sure to split your logic into modules**
- Please make sure that you seperate your logic in functions
- Make the names of your functions and variables are expressive of what they are
- Start by defining each functionality and the steps you should make to achieve it
- Please make sure not to pollute the global scope[refrain from using global variables]

- Before writing a function, ask yourself, is there a function in javascript that can do that, if not, write your own.

**Tips:**

- If you can't do something, search, then ask for help, this is not an exam.
- Please please please, don't copy someone else's code.

**Useful Links:**

- npm | build amazing things
- Nodejs Docs
- Fs module docs fs (you will need to use the sync version of the functions)[ends with the word Sync]
- HTTP module to make your http server
- Nodemon for restarting node
- Headers - Web APIs
- Introduction to Node.js

Please send the labs to either, please write your name and lab01 in the title of the email, also the name of the folder and I'll reply to each e-mail with the notes and comments.
OR email me the github link, make sure to ignore the node module folder
--
mohamedgomran+sdmans41@gmail.com

**Useful Readings:**

- Build an HTTP Server
- An introduction to the npm package manager
- The V8 JavaScript Engine
- How the Web works - Learn web development

**Must read before Day3:**

- Introduction to Mongoose for MongoDB
- REST API Best Practices – REST Endpoint Design Examples
- Authorization vs Authentication
- Hashing vs. Encryption vs. Encoding
- JWT
- ES6