**Mini test**

## 1. Mục tiêu

* Hiểu được các kiến thức cơ bản về thư viện Express.js
* Hiểu và vận dụng được cách cài đặt môi trường phát triển Express.js
* Hiểu và vận dụng được cách cài đặt các package cơ bản như nodemon, body-parser… để tạo lập một web server đầy đủ tính năng cơ bản với Express.js
* Hiểu được khái niệm và nắm được cách routing cơ bản và nâng cao trong Express.js.
* Hiểu và nắm được cơ chế hoạt động của Middleware trong Express.js.
* Hiểu và nắm được cơ chế serving static file trong Express.js.
* Hiểu và nắm sự khác biệt giữa SSR (Server-side rendering) website vs CSL (Client-side rendering) website
* Hiểu được cách tạo một CSL dynamic website.

**2. Yêu cầu**

* Đảm bảo clean code, naming convention, coding convention
* Tạo project folder theo hướng dẫn sau
  + Tên folder [Name\_Class]\_Exam và cho toàn bộ file vào trong
  + VD: [BachNX\_JS07]\_Exam
* Sau khi hoàn thành project, đẩy code lên github và nộp đường link github cho người phụ trách
* Sử dụng Express.js cho back-end
* Sử dụng ReactJS và các công nghệ phục vụ cho phần giao diện (front-end)

## 3. Mô tả

## Exercise 01 (1 point)

Khởi tạo một web server cơ bản bằng Express.js

* Tạo folder “<Name\_Class>\_Exam" để lưu trữ toàn bộ file code server
* Tạo file app.js để chứa code server
* Cài đặt package nodemon, express, morgan
* Gửi về cho client câu chào “Hello world” tại localhost(http://127.0.0.1), cổng 3000

## Exercise 02 (5 point)

Tiến hành tải thư mục dev-data/ có chứa 2 file questions.json và backup-questions.json và cho vào thư mục “ask-community-project”

<https://drive.google.com/drive/folders/1UcIjxqxMxEPJ9svLr34matHSox238uYF?usp=sharing>

Trong file todos.json là một mảng chứa thông tin về các todos ngẫu nhiên

Xây dựng một api nhỏ để thực hiện các thao tác gửi

GET (Read)/ POST (Create)/ PUT (update)/ DELETE (delete) request đến endpoint và xử lý chúng

**NOTE:**

* Sử dụng POSTMAN để test api !!!
* Khi đọc file dạng “.json", data lấy về được phải sử dụng JSON.parse(data) để convert về dạng Javascript đọc được
* Khi ghi dữ liệu vào file “.json", trước khi được ghi vào phải sử dụng JSON.stringify(data) để convert data về dạng json.

**Bước 1: (1 point)**

Tạo GET request với đường dẫn “/api/v1/todos”

* Gõ GET request đến “/api/v1/todos” trong POSTMAN
* Lấy ra một mảng todos và trả về cho client dữ liệu dưới dạng json

**Bước 2: (1 point)**

Tạo GET request với đường dẫn “/api/v1/todos/:id”

* Gõ GET request đến “/api/v1/todos/${id}” trong POSTMAN
* Tiến hành tìm kiếm todo trong mảng todos thông qua id
* Trả về cho client dữ liệu dưới dạng json todo vừa tìm kiếm được

**Bước 3: (1 point)**

Tạo POST request với đường dẫn “/api/v1/todos”

* Gõ POST request đến “/api/v1/todos” trong POSTMAN
* Tiến hành tìm kiếm todo theo trường title
  + Nếu không tìm thấy todo → tiến hành thêm mới dữ liệu từ client và trả về cho client dữ liệu dưới dạng json message {message: “Create successfully” }
  + Nếu tìm thấy todo → trả về cho client dữ liệu dưới dạng json message {message: “Todo already exists” }

**Bước 4: (1 point)**

Tạo PUT request với đường dẫn “/api/v1/todos/:id”

* Gõ PUT request đến “/api/v1/todos” trong POSTMAN
* Tiến hành tìm kiếm todo theo trường id
  + Nếu không tìm thấy todo → trả về cho client dữ liệu dưới dạng json message {message: “Todo not found” }
  + Nếu tìm thấy todo → tiến hành update dữ liệu gửi từ client trả về cho client dữ liệu dưới dạng json message {message: “Update successfully” }

**Bước 5: (1 point)**

Tạo DELETE request với đường dẫn “/api/v1/todos/:id”

* Gõ DELETE request đến “/api/v1/todos” trong POSTMAN
* Tiến hành tìm kiếm todo theo trường id
  + Nếu không tìm thấy todo → trả về cho client dữ liệu dưới dạng json message {message: “Todo not found” }
  + Nếu tìm thấy todo → tiến hành xóa dữ liệu và trả về cho client dữ liệu dưới dạng json message {message: “Delete successfully” }

## Exercise 03 (1 point)

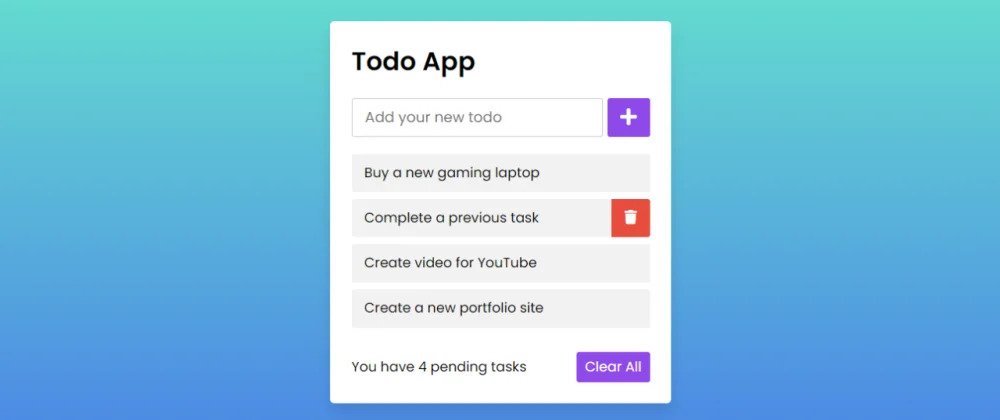
* Viết một middleware tên là checkExist
  + Middleware này sẽ được truyền vào toàn các Route (GET:id/POST/PUT/DELETE) ở exercise 02
  + Middleware này có tác dụng check nếu todo có tồn tại hay không theo cả id và title
  + Đối với trường hợp id
    - Nếu không tìm thấy todo → trả về cho client dữ liệu dưới dạng json message {message: “Todo not found” }
    - Nếu tìm thấy todo → gọi hàm next() để chạy đến callback function cuối cùng
  + Đối với trường hợp title
    - Nếu tìm thấy title → trả về cho người dùng dưới dạng json message {message: “Todo already exists” }
    - Nếu không tìm thấy todo → gọi hàm next() để chạy đến callback function cuối cùng
* NOTE: Xoá logic kiểm tra tồn tại todo của các route trên để truyền middleware checkExist vào sử dụng

## Exercise 04 (4 points + bonus)

Sử dụng các kiến thức đã học, xây dựng một todo list app với đầy đủ các tính năng

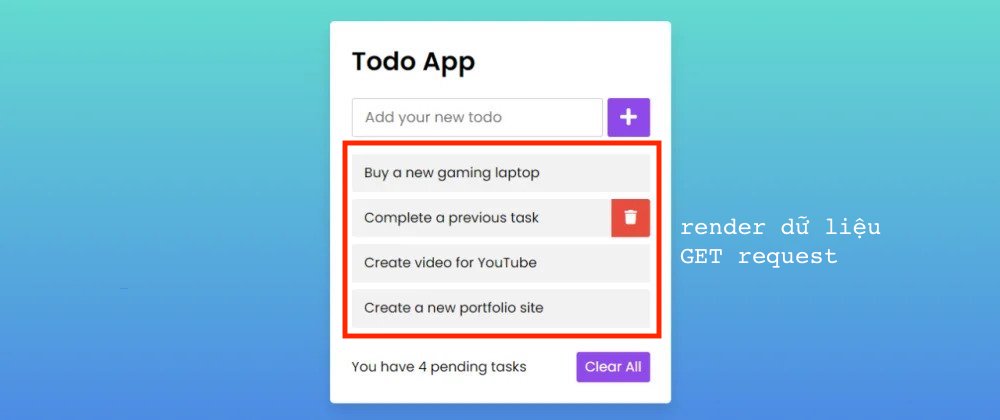
GET (Read)/ POST (Create)/ PUT (update)/ DELETE (delete)

* Vẫn trong thư mục “<Name\_Class>\_Exam" lưu trữ toàn bộ file code server
* Tạo đường dẫn “/” để render ra file HTML có giao diện như bên dưới
* Giao diện có thể tùy biến nhưng vẫn phải hài hoà màu sắc và font chữ
* Tiến hành thực hiện chức năng cho todo list app

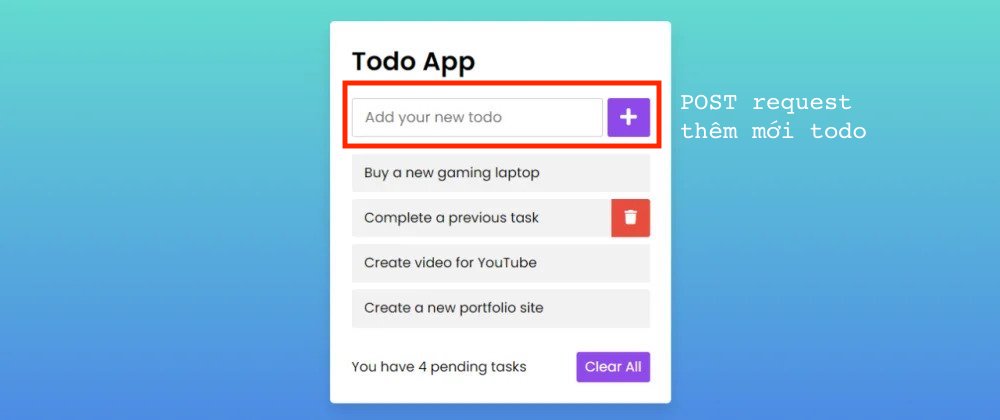


*Giao diện ứng dụng*

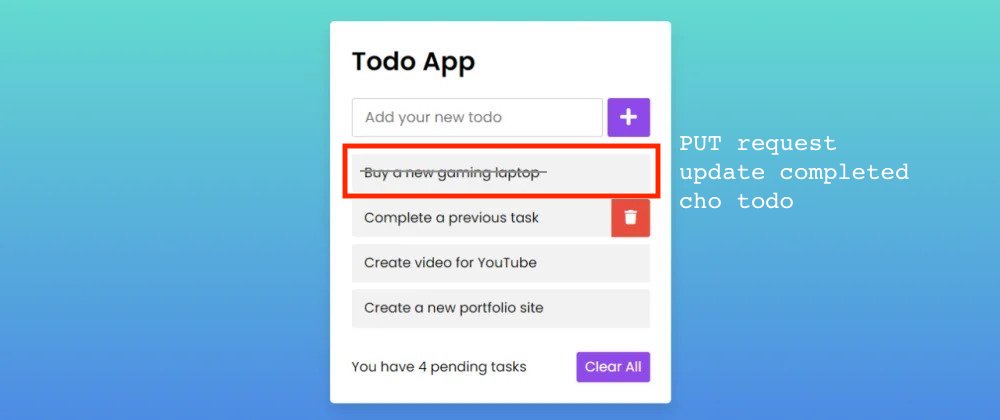
* GET (Read) (1 point)
  + Trong GET request với đường dẫn “/api/v1/todos” thêm query string “?per\_page=number” khi fetch với number là số lượng bản ghi todo muốn lấy ra từ file todos.json
  + Tiến hành sử dụng fetch gửi GET request lên server
  + Tại server, lấy query string “per\_page”, đọc file todos.json, lấy ra “number” bản ghi đầu tiên và gửi về cho client
  + Tiến hành render ra các todo giống với giao diện bên dưới
  + Với todo nào có trường “completed: true” → phải thêm dấu gạch chéo vào text
  + Đếm số lượng number todo có trường “completed: false” → render ra dòng chữ “You have number pending task"



* POST (Create) (1 point)
  + Thực hiện chức năng thêm mới todo
  + Todo mới phải được thêm vào đầu tiên của mảng dữ liệu cũng như của DOM
  + Khi thêm mới vào DOM phải đếm lại số lượng number todo có trường “completed: false” → render ra dòng chữ “You have ${number} pending task"



* PUT (Update) (1 point)
  + Thực hiện chức năng update trạng thái “completed” cho todo
  + Trạng mới phải được update trên DOM với đoạn title bị gạch ngang và update cả mảng dữ liệu trên server
  + Khi thay đổi trạng thái “completed” trong DOM phải đếm lại số lượng number todo có trường “completed: false” → render ra dòng chữ “You have ${number} pending task"



* DELETE (Delete) (1 point)
  + Thực hiện chức năng delete cho mỗi todo
  + Mỗi khi hover vào mỗi todo trên DOM phải hiện ra một button delete
  + Khi click vào button delete phải xoá dữ liệu ở trên server
  + Sau khi xóa xong dữ liệu ở trên server, tiến hành sử dụng window.location.href = “/” để refresh lại trang chủ để render lại {number} bản ghi đầu tiên

**BONUS points**

* DELETE ALL (Delete) (1 point)
  + Tạo thêm một DELETE request với đường dẫn “/api/v1/todos”
  + Thực hiện chức năng delete all cho button “clear all"
  + Khi click vào button delete phải xoá toàn bộ dữ liệu ở trên server
  + Sau khi xóa xong dữ liệu ở trên server, tiến hành sử dụng window.location.href = “/” để refresh lại trang chủ để render lại {number} bản ghi đầu tiên
* Thêm loading cho mỗi lần fetch (1 point)

## 4. Đánh giá

Để hoàn thành bài tập, học viên cần:

* Đưa mã nguồn lên GitHub.
* Dán link của repository lên phần nộp bài trên hệ thống.