

In the name of Allah

We first create a simple HTML file like "upload.html" to implement the requested problem.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Upload File</title>
  </head>
  <body>
    <form
      enctype="multipart/form-data"
      action="http://localhost:8080/upload"
      method="post"
    >
      <input type="file" name="myFile" />
      <input type="submit" value="upload" />
    </form>
  </body>
</html>
```

The first step) Post request

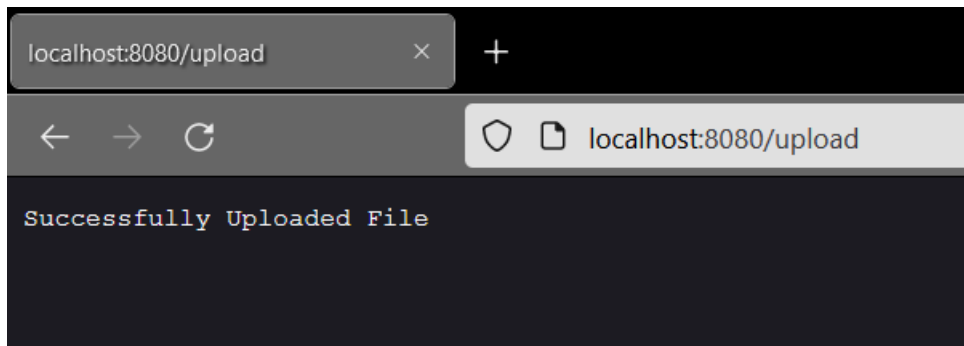
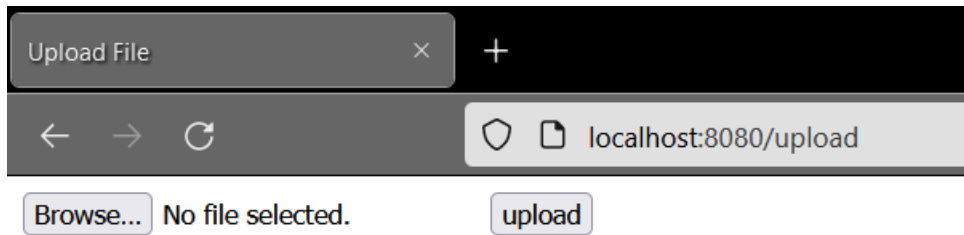
The user will go to the file submission page if he enters the following URL.

<http://localhost:8080/upload>

According to the figure below



After selecting the desired file from the system and pressing the upload button, the file will be uploaded to the webserver. If successful, the webserver will print the following message.



Our main code starts reading the uploaded file here and behind the scenes. We first store the values attributed to the main key separately and keep it as the number one rectangle.

Next, we start reading the values attributed to the input key with a loop.

We save each line we read from the input as a number two rectangle.

We compare the two rectangles, 1 and 2, according to the following function. If these two rectangles have overlapped, we save the rectangle attributed to the input key in a new structure containing the file's upload time. (Of course if the width and heights have positive values, then we call isRectanglesOverlap function)

```
func isRectangleOverlap(rec1 []int, rec2 []int) bool {  
    x1, x2, y1, y2 := rec1[0], rec1[0]+rec1[2], rec1[1], rec1[1]+rec1[3]  
    x12, x22, y12, y22 := rec2[0], rec2[0]+rec2[2], rec2[1], rec2[1]+rec2[3]  
    return x1 < x22 && x12 < x2 && y1 < y22 && y12 < y2  
}
```

This process is performed until the end of the values attributed to the input key.

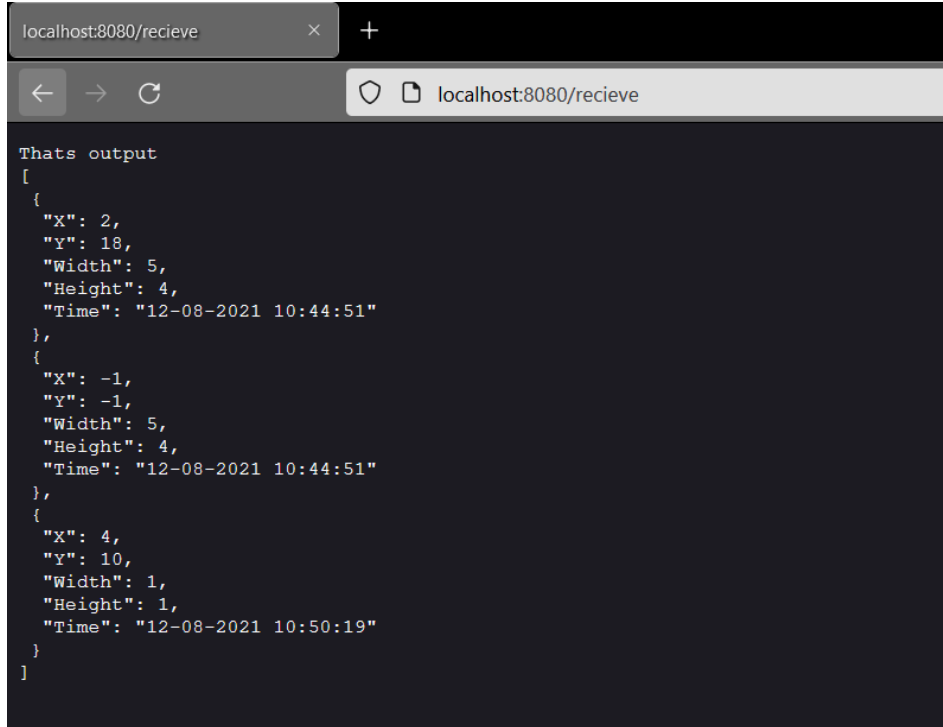
We considered a simple JSON file stored in the program folder for storage("./test.json"). We first read the file in each step and then add new values at the end.

Part 2) Get

In this section, the user, for correct rectangles, must enter the following address in his browser.

<http://localhost:8080/recieve>

Here, we wrote another function that goes and reads the values stored in the JSON file mentioned above and prints the values on the browser page according to the defined structure.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/recieve'. The page content shows the text 'Thats output' followed by a JSON array of three objects. Each object contains properties for 'X', 'Y', 'Width', 'Height', and 'Time'.

```
Thats output
[
  {
    "X": 2,
    "Y": 18,
    "Width": 5,
    "Height": 4,
    "Time": "12-08-2021 10:44:51"
  },
  {
    "X": -1,
    "Y": -1,
    "Width": 5,
    "Height": 4,
    "Time": "12-08-2021 10:44:51"
  },
  {
    "X": 4,
    "Y": 10,
    "Width": 1,
    "Height": 1,
    "Time": "12-08-2021 10:50:19"
  }
]
```