**PROJECT TITLE**: Task Manager

**Summary:**

A simple Task Manager application that allows a user to keep track of his or her daily tasks. The user should be able to categorise and prioritise their tasks based on predefined priorities and labels (Critical, High, Medium, Low).

**Responsibilities:**

- Created RESTful services using spring boot
- Developed logical part requirements using Java

**Functionalities:**

- **Create**

This create operation is used to create a Task and Label and stores in database.

localhost:7891/task/api/v1/create- creates task

localhost:7891/label/api/v1/create- creates label

Status code:

200 OK (Successful)

401 (Unauthorized Access)

500 (Internal Server Error)

400 (Bad Request)

Example:

Request:

```
{
    "task":"play footbaLL",
    "label":{
       "labelId":5
    },
    "priority":"LOW",
    "date":"2022-05-08"
}
```

Response:


```
{
    "data": {
        "taskId": 15,
        "task": "play footbaLL",
        "label": {
            "labelId": 5,
            "labelName": null
                },
        "priority": "LOW",
        "date": "2022-05-08
            },
    "error": false
}
```


•**Read**:

This operation is used to fetch the tasks and label from the database based on task ID and label ID.

GET:

localhost:7891/task/api/v1/getAllTask

localhost:7891/label/api/v1/getAllLabel

Status Code:

200 ok (Successful)

401(Unauthrozied Access)

500(Internal Server Error)

400(Bad Request)

Example:

Response:


```
{
    "data": [
        {
            "labelId": 1,
            "labelName": "EDUCATION"
        },
        {
            "labelId": 2,
            "labelName": "ENTERTAINMENT"
        },
        {
            "labelId": 3,
            "labelName": "WORK"
        },
        {
            "labelId": 4,
            "labelName": "TRAVEL"
        },
        {
            "labelId": 5,
            "labelName": "SPORTS"
        },
        {
            "labelId": 14,
            "labelName": "TODO"
        }
    ],
    "error": false
}
```


•**Update**:

By using this operation, we can update task and label based on task ID and label ID.

PUT:

localhost:7891/task/api/v1/update

localhost:7891/label/api/v1/update

Status Code:

200 ok (Successful)

401(Unauthorized Access)

500(Internal Server Error)

400(Bad Request)

Example:

Request:


```
{
    "labelId":5,
    "labelName":"SPORTS"
}
```


Response:


```
{
    "data": {
        "labelId": 5,
        "labelName": "SPORTS"
    },
    "error": false
}
```


•**Delete**:

This operation is used to delete the task and label permanently from the data base.

DELETE:

localhost:7891/task/api/v1/delete

localhost:7891/label/api/v1/delete

Status Code:

200 ok (Successfull)

401(Unauthrozied Access)

500(Internal Server Error)

400(Bad Request)

Example:

Request:

localhost:7891/task/api/v1/remove?id=15

Response:


```
{
    "data": "deleted successfully",
     "error": false
}
```

**•Finding next two days task:**

In this operation the user can find the next two days task from the current date.

GET:

localhost:7891/task/api/v1/nextTwoDaysTask

Status Code:

200 ok (Successfull)

401(Unauthrozied Access)

500(Internal Server Error)

400(Bad Request)

Example:

Response:


```
{
    "data": [
        "watch movie"
    ],
    "error": false
}
```


**•Finding today's task of particular label:**

In this operation the user can find the current day task based on particular label by giving labelId

GET:

localhost:7891/task/api/v1/currentDayTask

Status Code:

200 ok (Successfull)

401(Unauthrozied Access)

500(Internal Server Error)

400(Bad Request)

Example:

Response:


```
{
    "data": {
        "taskId": 17,
        "task": "play kabadi",
        "label": {
            "labelId": 5,
            "labelName": null
        },
        "priority": "LOW",
        "date": "2022-04-24"
    },
    "error": false
}
```


**•Finding all Critical and high priority tasks:**

In this operation the user can view all the high priority and critical tasks.

GET:

localhost:7891/task/api/v1/criticalAndHighTask

Status Code:

200 ok (Successfull)

401(Unauthrozied Access)

500(Internal Server Error)

400(Bad Request)

Example:

Response:


```
{
    "data": [
        "study java",
        "study python",
        "apply for pancard",
        "goa trip"
    ],
    "error": false
}
```