

CitrusLeaf Software

Python Developer | Coding Task

Task:

Create REST API using Python/Flask with MySQL as DB, generate a Postman collection for the same

- Use the .env file to store database credentials.
- Authentication: Create a login and register APIs. Register with Fields: name, email, password. Login with email/password. Passwords have to be encrypted before saving in the db.
- Use JWT token for authorization
- Task CRUD: (fields: title, due date, attachment (file), user)
 - Create Task API- Create a task with the above-mentioned fields and assign it to the current logged-in user with it in the DB.
 - Task list API- get tasks of the logged-in user
 - Edit Task- similar to create a task but on a given task ID
 - Delete Task- delete a task by ID.
- Only logged in users can perform the CRUD operations for his/her tasks.
- Upload a CSV file for multiple tasks and import it to the database.

This project provides a comprehensive REST API for efficient task management with secure user authentication and authorization. It leverages Flask and MySQL to deliver a scalable and maintainable solution.



















Technologies Used :

- Python
- Flask
- MySQL
- Visual Studio Code
- JWT
- Werkzeug
- Postman


















Screenshots :

Database Table Structure :

User table:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/> 2	public_id 	varchar(255)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 3	name	varchar(20)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 4	email 	varchar(20)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/> 5	password	varchar(255)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More

Task table :

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/> 2	title	varchar(50)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 3	due_date	datetime			Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 4	attachment	varchar(255)	utf8mb4_general_ci		Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 5	user_id 	int(11)			No	None			 Change  Drop  More

Task Management API :

User Registration

User registration to add new user to the User table. The password will be hashed and saved to the database.

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:5000/user/register`
- Method:** `POST`
- Body (JSON):**

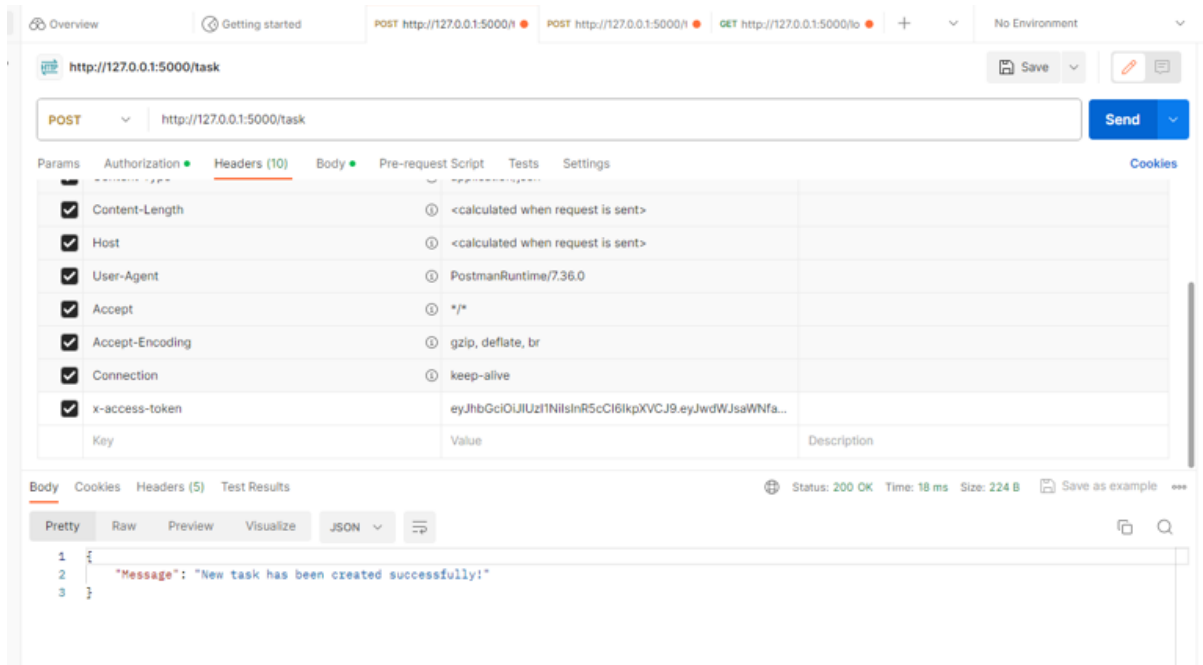
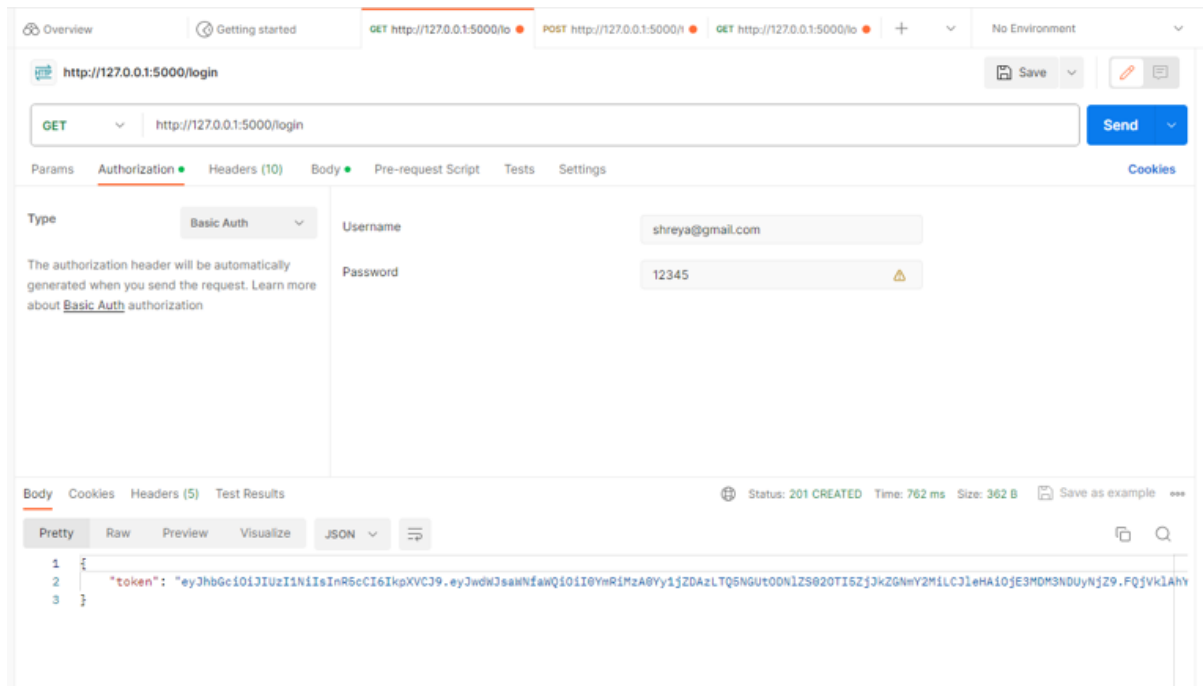
```
{  1: {  2:   "name": "Shivam",  3:   "email": "shivam@gmail.com",  4:   "password": "123456"  5: }
```
- Status:** 200 OK, Time: 689 ms, Size: 214 B
- Response Body (JSON):**

```
{  1: {  2:   "Message": "New user has been registered!"  3: }
```

		id	public_id	name	email	password
<input type="checkbox"/>	Edit Copy Delete	2	802b2f76-8361-4513-aa26-71c8cb38b8f3	Dhiraj	dhiraj@gmail.com	pbkdf2:sha256:600000
<input type="checkbox"/>	Edit Copy Delete	3	1fd85767-237a-4d9a-8cc7-fbeedae8b4fe	Pragati	pragati@gmail.com	pbkdf2:sha256:600000
<input type="checkbox"/>	Edit Copy Delete	4	7ebda9f1-eaf9-4c0f-bdbd-006303741a4e	Suraj	suraj@gmail.com	pbkdf2:sha256:600000
<input type="checkbox"/>	Edit Copy Delete	5	4bdb304c-cd03-494e-83ee-6929f2ddcfcc	Shreya	shreya@gmail.com	pbkdf2:sha256:600000\$DNEXCcDoWlleY2CS\$208c6094edac...
<input type="checkbox"/>	Edit Copy Delete	6	84217cc6-6152-4d0e-8a38-de70551eba04	Shiva	shiva@gmail.com	pbkdf2:sha256:600000\$0IDXOQOdkmbDAoGs\$f2d77c8d3528...
<input type="checkbox"/>	Edit Copy Delete	7	244a25c7-f7a8-4615-8d0f-16cd74320a4d	Shivam	shivam@gmail.com	pbkdf2:sha256:600000\$WbmRQqd5NVKtVJB4\$eac6e2eeb30a...

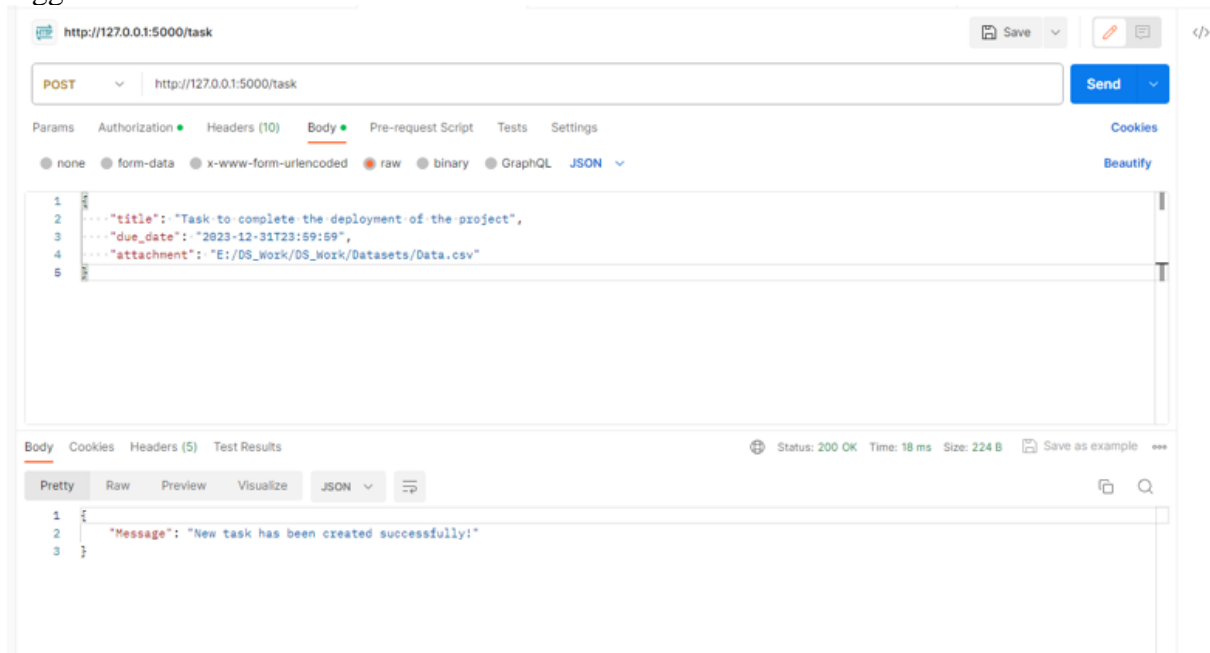
Login | User Authorization using JWT token :

When user login to the system, token get generated, and the session starts for 30 minutes. Till the user is login can perform different operations on Tasks, to create a new task,



Create new Task :

Logged in user can create a new task, the task will be saved to the database assigned to the logged in user with User ID.



`SELECT * FROM `task``

☐ Profiling [\[Edit inline \]](#) [\[Edit \]](#) [\[Explain SQL \]](#) [\[Create PHP code \]](#) [\[Refresh \]](#)

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	id	title	due_date	attachment	user_id
<input type="checkbox"/> Edit Copy Delete	1	Task to complete the deployment of the project	2023-12-31 23:59:59	E:/DS_Work/DS_Work/Datasets/Data.csv	5
<input type="checkbox"/> Edit Copy Delete	2	Update on current project meet with Client	2023-12-30 23:59:59	E:/DS_Work/DS_Work/Datasets/Predict_BMI.csv	5

Retrieve Tasks :

User can get the details of the Tasks assigned to him/her. The Tasks list will get displayed.

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:5000/my_tasks`. The response is a JSON array of two task objects. The status is 200 OK, and the response size is 595 B.

```
1 {
2   "tasks": [
3     {
4       "attachment": "E:/DS_Work/DS_Work/Datasets/Data.csv",
5       "due_date": "Sun, 31 Dec 2023 23:59:59 GMT",
6       "id": 1,
7       "title": "Task to complete the deployment of the project"
8     },
9     {
10      "attachment": "E:/DS_Work/DS_Work/Datasets/Predict_BMI.csv",
11      "due_date": "Sat, 30 Dec 2023 23:59:59 GMT",
12      "id": 2,
13      "title": "Update on current project meet with Client"
14    }
15  ]
16 }
```

Edit or Update the Tasks :

User can update the task by fetching the task by task ID.

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:5000/task/1`
- Method:** `PUT`
- Body (JSON):**

```
{
  "title": "Deployment is completed: Client Meet for updates",
  "due_date": "2023-12-31T23:59:59",
  "attachment": "Project_details.csv"
}
```
- Status:** 200 OK, Time: 21 ms, Size: 206 B
- Response (JSON):**

```
{
  "Message": "Task has been updated"
}
```

The task is updated with updating the Title and the datetime as

The screenshot shows a REST client interface with the following details:

- URL:** `http://127.0.0.1:5000/my_tasks`
- Method:** `GET`
- Status:** 200 OK, Time: 13 ms, Size: 580 B
- Response (JSON):**

```
{
  "tasks": [
    {
      "attachment": "Project_details.csv",
      "due_date": "Sun, 31 Dec 2023 23:59:59 GMT",
      "id": 1,
      "title": "Deployment is completed: Client Meet for updates"
    },
    {
      "attachment": "E:/DS_Work/DS_Work/Datasets/Predict_BMI.csv",
      "due_date": "Sat, 30 Dec 2023 23:59:59 GMT",
      "id": 2,
      "title": "Update on current project meet with Client"
    }
  ]
}
```

Delete Task :

User can delete the task by ID

The screenshot shows a REST client interface with the following details:

- URL: `http://127.0.0.1:5000/task/1`
- Method: `DELETE`
- Body: Empty
- Status: `200 OK`, Time: `19 ms`, Size: `211 B`
- Response Body (JSON):

```
{  "Message": "The task has been deleted:"}
```

The screenshot shows a database management interface with the following details:

- SQL Query: `SELECT * FROM `task``
- Table Name: `task`
- Columns: `id`, `title`, `due_date`, `attachment`, `user_id`
- Table Data:

id	title	due_date	attachment	user_id
2	Update on current project meet with Client	2023-12-30 23:59:59	E:/DS_Work/DS_Work/Datasets/Predict_BMI.csv	5