Task Management Application Documentation Backend

Git Url - https://github.com/sumeshjr/TaskApp/tree/master

Introduction

The Task Management Application is a web-based system developed using Django, a high-level Python web framework. This application provides users with the ability to register, log in, and manage tasks effectively. The system employs Django's Model-View-Controller (MVC) architecture, and the data is stored in a relational database.

Technologies Used

Django: The web framework used for backend development.

Django REST Framework: An extension for Django to facilitate the creation of RESTful APIs.

MongoDB Cloud: The cloud-based NoSQL database for storing user and task data.

Models

UserLogin Model

The `UserLogin` model represents user information for registration and login.

- `id`: Auto-incremented primary key.
- `username`: CharField for storing the username.
- `password`: TextField for storing the user's password.

Task Model

The `Task` model represents individual tasks associated with users.

- `userid`: IntegerField representing the user ID associated with the task.
- `name`: CharField for storing the task name.
- `description`: TextField for storing the task description.
- `priority`: CharField for storing the task priority.
- `date`: DateField for storing the task deadline.
- `is_completed`: BooleanField indicating whether the task is completed.

Serializers

UserSerializer

Serializes the `UserLogin` model for API interactions.

TaskSerializer

Serializes the `Task` model for API interactions.

Views

User Registration

Endpoint: `/reg/` (POST)

- Description: Registers a new user with a unique username and password.

- Request Body:

  ```json
  {
    "username": "unique_username",
    "password": "user_password"
  }
  ```

- Response:

  - Success:

    ```json
    {
      "status": "Success"
    }
    ```

  - Failure:

    ```json
    {
      "status": "Failed"
    }
    ```

User Login

Endpoint: `/log/` (POST)

- Description: Logs in an existing user.
- Request Body:

  ```json
  {
    "username": "existing_username",
    "password": "user_password"
  }
```

```
- Response:
  - Success: Returns user data including the user ID, username, and password.
  - Failure: Returns an empty list if the username and password do not match any user.

**Task Management**

**Add Task**

Endpoint: `/add/` (POST)

- Description: Adds a new task for a specific user.
- Request Body:
  ```json
  {
    "userid": 1,
    "name": "Task Name",
    "description": "Task Description",
    "priority": "High",
    "date": "2023-12-01",
    "is_completed": false
  }
  ```
- Response:
  - Success:
    ```json
    {
      "status": "Success"
    }
    ```
  - Failure:
    ```json
```
```

```
{
  "status": "Failed"
}
```

View All Tasks

Endpoint: `/view/` (POST)

- Description: Retrieves all tasks in the system.
- Request Body: Empty
- Response: Returns a list of tasks.

Update Task

Endpoint: `/update/` (POST)

- Description: Updates the name of a specific task.
- Request Body:
```json
{
  "userid": 1,
  "name": "New Task Name"
}
```
- Response:
  - Success:
```json
{
  "status": "Success"
}
```

- Failure: Returns an empty list if the user or task is not found.

Delete Task

Endpoint: `/delete/` (POST)

- Description: Deletes a specific task.
- Request Body:

```json
{
  "taskid": 1
}
```

- Response:
  - Success:

```json
{
  "status": "Success",
  "message": "Task deleted successfully"
}
```

  - Failure: Returns an error message if the task is not found or an internal server error occurs.

View User's Tasks

Endpoint: `/mytask/` (POST)

- Description: Retrieves all tasks for a specific user.
- Request Body:

```json
{
  "userid": 1
```

}
```

- Response: Returns a list of tasks for the specified user.

**Search Tasks**

Endpoint: `/search/` (POST)

- Description: Searches for tasks based on a keyword in the name or description.
- Request Body:

```json
{
  "search_keyword": "keyword"
}
```

- Response:
  - Success: Returns a list of tasks matching the search criteria.
  - Failure: Returns an error message if an internal server error occurs.

Update Task Status

Endpoint: `/update/` (POST)

- Description: Updates the completion status of a specific task.
- Request Body:

```json
{
  "task_id": 1,
  "is_completed": true
}
```

- Response:
```

- Success:

  ```json
  {
    "success": true,
    "message": "Task updated successfully"
  }
  ```

- Failure: Returns an error message if the task is not found or an internal server error occurs.

Conclusion

The Task Management Application provides a robust solution for users to manage their tasks efficiently. It leverages the Django framework and RESTful API principles to enable seamless interactions between the frontend and backend. The documentation serves as a guide for users and developers to understand and utilize the application's features effectively.