Test task

Build a simple API to manage users and their tasks. The API should be built using PHP and should adhere to RESTful principles.

Our team is using Yii Framework 2.x and MongoDB as the main database. For this task, **you can use any framework and database you like**. Using a stack similar to ours will be a plus.

The final result should be sent to HR as an archive or as a link to the public GitHub/GitLab/BitBucket repository.

User data requirements

Fields:

- Login (minimum of 4 characters, unique)
- Password (minimum of 6 characters)
- First name (first letter required to be uppercase)
- Last name (first letter required to be uppercase)
- E-mail address (unique)
- Registration date and time (should be set automatically when the user created)

All fields are mandatory.

User task data requirements

Fields

- Title (non-empty string)
- Description (non-empty string)
- Status (allowed values are: New, In Progress, Failed, Finished)
- Start date and time

All fields are mandatory.

API requirements

- Use JSON for a data exchange format.
- Only unprocessed (with status “New”) tasks are allowed to be deleted.
- All date fields should be formatted to “DD-MM-YYYY HH:mm” in API responses.
- Users should be able to create a user by sending a POST request to the API endpoint `/users`
- Users should be able to retrieve a list of users by sending a GET request to the API endpoint `/users`. The API should return a JSON response with an array of users.
- Users should be able to retrieve a single user by sending a GET request to the API endpoint `/users/{id}`, where `{id}` is the ID of the user. The API should return a JSON response with the details of the user.
- Users should be able to update a user by sending a PUT request to the API endpoint `/users/{id}`, where `{id}` is the ID of the user.
- Users should be able to delete a user by sending a DELETE request to the API endpoint `/users/{id}`, where `{id}` is the ID of the user. All related user tasks should be removed.
- Users should be able to create a task by sending a POST request to the API endpoint `/users/{id}/tasks`, where `{id}` is the ID of the user.
- Users should be able to retrieve a list of user tasks by sending a GET request to the API endpoint `/users/{id}/tasks`, where `{id}` is the ID of the user. The API should return a JSON response with an array of tasks.
- Users should be able to retrieve a single task by sending a GET request to the API endpoint `/users/{id}/tasks/{task-id}`, where `{id}` is the ID of the user and `{task-id}` is the ID of the task. The API should return a JSON response with the details of the task.
- Users should be able to update a task by sending a PUT request to the API endpoint `/users/{id}/tasks/{task-id}`, where `{id}` is the ID of the user and `{task-id}` is the ID of the task.
- Users should be able to delete an unprocessed task by sending a DELETE request to the API endpoint `/users/{id}/tasks/{task-id}`, where `{id}` is the ID of the user and `{task-id}` is the ID of the task.
- Users should be able to delete all unprocessed tasks by sending a DELETE request to the API endpoint `/users/{id}/tasks`, where `{id}` is the ID of the user.

- Users should be able to get statistics of how many tasks per each status are on a particular user by sending a GET request to the API endpoint `/users/{id}/tasks/stats`
- Users should be able to get statistics of how many tasks per each status are in general among all users by sending a GET request to the API endpoint `/tasks/stats`
- Add pagination to next endpoints: `/users`, `/users/{id}/tasks`
- Add sorting to the `/users` endpoint by fields First name, Last name, and E-mail.
- Add sorting to the `/users/{id}/tasks` endpoint by fields Title and Status.

Implementation requirements

- Ensure proper error handling.
- Provide clear instructions on how to use the API.
- *(Optional, will be a plus)* Write unit tests to test the API's functionality.