Project: Task Management Tool (Trello/Clickup clone)

Description:

Allows users to create tasks, assign users who will work on them, monitor and update status of tasks.

Features:

- Authentication
  - Login
  - Register
  - Verification after registration via e-mail
- Create different projects which will hold their respective tasks.
- Tasks
  - Create / Update / Delete task.
  - Should allow adding attachments to a task.
  - Assign users to it.
  - Update status (Unassigned, Open, WIP, etc). Should be able to create a new status if required.
  - Comments (will allow users to specify updates if required on that task).
  - View, search and filter tasks.
  - Notify users if they get added to some task or there is any change to the task.

Additional Features (Good to have):

- Task description should allow formatting options (checklist, numbered list, bold, underline, etc.)
- View / Download a report which displays each task (can be filtered) its details and stats (time taken on each stage or fully complete the task).
- Drag / Drop tasks to move between statuses.

For UI Reference Check

- Clickup (https://app.clickup.com/)
- Trello (https://trello.com/)

Stack:

- HTML
- CSS (Can use one framework from below or any other)
  - https://getbootstrap.com/
  - https://mdbootstrap.com/docs/standard/getting-started/installation/
  - https://bulma.io/
- PHP and Laravel framework
- Vanilla Javascript or jQuery
- MySQL
- Mailtrap.io (Use for mocking an email service)

Implementation:

- Expose a proper REST API with authentication
- Add proper validations for each api fields and return message to user on errors
- Notify any server side errors to developer
- Keep UI and API decoupled (UI should call APIs via AJAX with proper authentication)
- Normalized database schema with appropriate indexes
- Long running tasks if any should be done asynchronously
- Responsive UI and good user experience (UI should be presentable like a complete product)
- Prevent any XSS/SQL Injections

Code

- Code should be well structured
- Consistent code styling and follow naming conventions (eg. Uppercase with underscores for constants, camel case for variables and function names etc.)
- Segregation of responsibilities in MVC pattern (Controllers for validations, Service classes for business logic and Models for queries)
- Can use any other design pattern if applicable
- Make use of framework features to keep code maintainable and readable (eg, use Form Requests from laravel to separate out validation logic into its own class for each endpoint)
- Error handling should be done in laravel's error handler instead of adding a try catch everywhere