🧩 Laravel Interview Project — TaskFlow

Project Description

TaskFlow is a lightweight Laravel-based project and task management application designed to showcase a developer's mastery of the Laravel framework.

The application allows users to create and manage projects, assign tasks, comment on tasks, and receive email notifications when tasks are updated. It leverages Laravel features such as **events, listeners, observers, notifications, collections, policies, console commands, and seeders**.

TaskFlow is fully **Dockerized** using Laravel Sail and integrates **Mailhog** for capturing email notifications. Console reporting includes both **terminal tables** and **Excel exports** via **Laravel Excel**, demonstrating practical, real-world Laravel development skills.

This project serves as a comprehensive assessment tool for Laravel developers, covering both backend functionality and developer workflow best practices, including automated setup via **Makefile** and clear documentation.

🎯 Objective

Build a small but production-style **Laravel web application** that demonstrates your ability to use Laravel effectively and professionally.

This assignment evaluates your knowledge of:

- Laravel architecture (Models, Controllers, Views)
- Events, Listeners, Observers, Notifications
- Collections and Eloquent relationships
- Console commands (terminal tables + Excel export)
- Authorization (Policies)
- Dockerized development with Laravel Sail
- Mailhog for email notifications
- Developer automation (Makefile)
- Seeders using Faker for demo data
- Documentation and commit practices

⚠️ Use **`web.php` routes** and **Blade templates** — not an API.
❌ No unit or integration tests required.
🧱 Use **normal Laravel authentication** (Breeze or Laravel UI).

💡 Scenario

You are building **TaskFlow**, a small internal project tracker.

Users can:

- Register and log in
- Create and manage projects
- Add and assign tasks
- Comment on tasks
- Receive email notifications when tasks are updated (captured via Mailhog)
- Seed demo data using Faker

🧰 Core Requirements

1. Authentication

Use standard Laravel authentication.
Authenticated users should see a `/dashboard` listing their projects.

---

2. Models & Relationships

Model	Key Fields	Relationships
User	name, email, password	hasMany(Project), hasMany(Task, 'assigned_to'), hasMany(Comment)
Project	title, description, owner_id	belongsTo(User), hasMany(Task)
Task	title, description, status, project_id, assigned_to	belongsTo(Project), belongsTo(User, 'assigned_to'), hasMany(Comment), hasMany(TaskHistory)
Comment	body, user_id, task_id	belongsTo(Task), belongsTo(User)
TaskHistory	task_id, event_type, old_values, new_values	belongsTo(Task)

---

3. CRUD (Blade + `web.php`)

Implement simple CRUD pages using **Blade templates** for:

- Projects (create, edit, view, delete)
- Tasks (create, assign, update, comment)
- Dashboard summary view

---

4. Observers

Implement a `TaskObserver` that logs all task lifecycle events (create, update, delete) into the `task_histories` table.

---

5. Events, Listeners & Notifications

Implement:

- **Event:** `TaskUpdated`
- **Listener:** `NotifyProjectMembers`
- **Notification:** `TaskUpdatedNotification`

When a task is updated:

- Dispatch `TaskUpdated`
- Notify the project owner and task assignee
- Send an email via Mailhog using the `toMail()` channel

---

6. Mailhog Integration

Include Mailhog in the Docker setup.

Mailhog web UI:
📫 http://localhost:8025

`.env` configuration:

```
MAIL_MAILER=smtp
MAIL_HOST=mailhog
MAIL_PORT=1025
MAIL_FROM_ADDRESS="noreply@taskflow.test"
MAIL_FROM_NAME="TaskFlow"
```

---

7. Collections

On each project's detail page, display task statistics using **Laravel Collections**:

- Total tasks
- Completed tasks
- Pending tasks
- Completion percentage

---

8. Policies

Implement a `ProjectPolicy`:

- Only project owners can edit/delete their projects
- Only project members can modify tasks

---

9. Console Commands

sail artisan report:tasks

Displays a project summary table in the terminal:

```
+------------+-------------+------------+-----------+----------------+
| Project    | Total Tasks | Completed  | Pending   | Completion %   |
+------------+-------------+------------+-----------+----------------+
```

sail artisan report:export

Exports the same report to Excel (`storage/app/reports/tasks_report.xlsx`)
Use **Laravel Excel** (`maatwebsite/excel`) for exporting.

---

10. Seeders

Implement seeders to generate demo data using **Faker** for: - Users - Projects - Tasks - Comments

Seeded data should allow the reviewer to explore the application without manual entry.

---

11. Docker / Laravel Sail

Use **Laravel Sail** for the development environment.

Required Docker services:

- `laravel.test` (PHP + Nginx)
- `mysql`
- `mailhog`

---

12. Makefile

Include a `Makefile` at the project root to simplify local development.

Required Make Targets:

- `up` – start containers
- `down` – stop containers
- `init` – install dependencies, copy `.env`, start Sail, run migrations & seeders
- `artisan` – run Artisan commands
- `reset` – rebuild and reinitialize a clean environment

🧩 Deliverables

You must:

1. Push the **completed project** to a **public GitHub repository**.
    - Use clear, descriptive commit messages.

- No GitFlow branching required — a clean, readable commit history is sufficient.

2. Include a clear **`README.md`** with setup instructions.

README Requirements

Your `README.md` must include:

- Project overview
- Prerequisites (Docker, Make, Composer)
- Setup steps (using the provided Make targets)
- Access URLs:
  - App → http://localhost
  - Mailhog → http://localhost:8025
- How to run console commands
- Example seeded credentials
- Notes on implemented features

Example Commands (for README reference)

```
make init
make artisan cmd="report:tasks"
make artisan cmd="report:export"
```

🧠 Evaluation Criteria

Category	Focus
Auth	Proper Laravel authentication
Routing	Correct use of `web.php` and middleware
Models	Proper Eloquent relationships
Observers	Task lifecycle logging

Category	Focus
Events & Listeners	Correct dispatching and handling
Notifications	Email notifications captured in Mailhog
Collections	Used effectively for summary stats
Console	Functional terminal table and Excel export via Laravel Excel
Seeders	Demo data using Faker is functional
Docker	Functional Sail setup
Makefile	Correct targets and developer usability
Documentation	Clear and accurate setup instructions
Git	Logical commits with meaningful messages

✅ Submission

- Push your completed project to a **public GitHub repository**
- Ensure the project can be started and seeded using:

```
make init
```

- Confirm that email notifications appear in **Mailhog** (`http://localhost:8025`)

🏁 Goal

The reviewer should be able to:

1. Clone the repository
2. Run one or two `make` commands
3. Access the working Laravel app and Mailhog
4. Verify the implementation of all Laravel features listed above

Good luck! This project demonstrates your **Laravel skills, workflow, and professionalism**.