Case Study

A company wants to automate its task allocation process. Currently, employees are assigned tasks either verbally or via email, and it is very hard to keep track of who is doing what, and how much is done. The company wants to move to a web-based task allocation system to do the following.

Anyone can create a task. Each task gets a corresponding ticket number. One user is assigned as "responsible" for the task, and others can be assigned as "helpers". Admins create and delete user accounts (individually, or in bulk via a CSV file).

When a task is created, by default the creator is "responsible" for it unless it is changed. Admins can also change the responsible person. The creator, or the current responsible person, can also change the assignment. The responsible person can add/remove helpers.

Responsible persons and helpers can post against a task. Once a task is complete, its status has to be changed to complete. Tasks can be in different statuses: Not initiated, initiated, progressing, stalled, waiting for resources, completed, unresolved being some of them.

When creating a task, the creator has to give a task title and a short description of no less than 50 words. He/She can assign the responsible person and/or helpers. He/she can also attach watchers to the task. All stakeholders, the creator, responsible person, helpers, and watchers should be notified with every post on the task.

Tasks can be dependent on other tasks, meaning they will never achieve the complete status, unless the other tasks are completed. Tasks can also have sub-tasks (not the same as a dependency), which can be created and assigned to other responsible people, but the responsible person on the main task will automatically become a watcher of the sub-tasks, as well as the tasks on which there is a dependency.

Task posts are mainly text but can have attachments. Special processing should be done on certain types of attachments. These are:

- Zip files and binary files should be scanned for viruses.
- Images should be limited to a maximum resolution of 1024 x 800. If necessary, the system should be resized on the fly.
- URLs should be stored as it is, but when displaying the URL, the title of the target HTML content should be rendered as well.
- Video files should be displayed in two parts:
    - The first part is a download link,
    - and the second part is a video player embedded in the view so that the viewer can play the video in the system itself.
- GIT URLs should be linked to the GIT repository as well as a small floating test showing how the repository can be cloned.
- When viewing PDF files should have a download link and an icon. Clicking on the icon should render the file on the system itself instead of downloading.