Chan Wai Hon Jonathan A0189909E

Use Cases (with possible future features):

User – wishes to use the app to keep track of tasks that need to be completed

- Create tasks
  - Task description (title)
  - Due date
  - Category
  - Priority
  - Comment
  - Subtask
  - Repeat?
  - **Reminder?**
- Update tasks
  - Destroy
  - Completed?
  - Update task details
- Read tasks
  - Filter
    - Category
    - Due date
      - All
      - Today
      - Next 7 days
    - Completed?
    - Priority
  - Sort
    - Priority
    - Due date
    - Name
- **Share tasks (When everything else is complete)**

Execution plan (with possible future features): (Highlighted indicates current progress)

Task	**Done**
Allow users to use basic tasks • C – new task with title (validate: present) • R – view task details • U – task details • D – task	Y
Allow users to use categories • C – name (validate: unique), associate with tasks (many-to-many), use previously created categories when creating tasks • R – filter by categories • U – task's categories	Y
Allow users to set due dates • C – due date (validate: future, present) • R – sort by date • U – due date	Y

Allow users to set completion • R – filter by completed • U - completed	Y
Allow users to choose from previously created categories • C – select from created categories in addition to creating new • U – preselect associated categories and allow user to add associations • D – categories (remove associations but not tasks)	Y
Allow users to add comments • C – comments • U – comments	Y
Allow users to indicate priority • C – priority • R – default sort by priority, include filter by priority level • U – priority	Y
Allow users to filter by due date • R – All (sort by due date then name), Today (default, sort by name), Week (sort by due date then name, divide into dates)	Y
Allow users to add subtasks • C – subtasks (similar features to task, but only title, completed fields) • U – subtask details • D – subtask	Y
Allow users to repeat daily tasks if completed (use cron) • C – repeat • U – repeat	Y
Try adding AJAX to site • Allow subtasks to be marked as completed from Show • Allow tasks to be marked as completed from Index	Y
Improve Subtask implementation • Add subtask from Task Show • Edit subtask from Task Show • Add validation for creating subtask from Task Show • Strikethrough when subtask is completed	
Make filter render listings on change.	
Add CSS to prettify everything • When moving between pages, there should not be significant changes in positioning. It should feel seamless.	
Refactor + Bughunt	
Allow users to sign up and log in	
Created tasks should be assigned to users	
Allow users to collaborate • C – creator and assigned to (assign tasks to other users) • R – tasks should be visible with all personal tasks, filter by assigned • U – task details (everyone can update)	

• D – only creator can delete tasks	
Users can only view their own / assigned tasks (require permissions)	
Improve functionality • Allow users to customize repeat frequency • Allow users to optionally set time (hide unless they want to add it)	

Problems:

1. With no experience in web development, it is difficult to tell whether my coding style is proper and whether I am doing things the traditional way.

2. Lack of background means that there is a lot of new information to process to understand how a web application works. To help with this learning I am trying not to rely on libraries, so that I can understand what is happening underneath instead of just knowing how to use Rails and its many helpful libraries.

3. After successfully implementing AJAX into a feature, I realised there might not actually be a need to use AJAX just because I can. I will have to reconsider my usages and decide whether AJAX will really improve UX as compared to just doing it via redirects. Will have to investigate the advantages of using AJAX. (E.g. Live filtering without submitting, live validations in a form)

4. In accordance to the DRY mindset, there are some parts that are partially repeated, i.e. the listings of tasks and subtasks. However, combining them into one might require a lot of conditionals which would overly complicated things, and appear as if I am forcing them together. Keeping them apart should make them easier to differentiate and work with.

5. I ended up with a lot of methods in the Model file. Should consider using a separate file to store these methods so that it will be more readable.

6. I encountered a HTML gotcha, e.g. forms not being allowed within table unless if it is entirely encapsulated within a cell. These will need to be fixed in the future. I will likely solve it by using CSS to give the form an appearance of a table instead of the method I am using right now (putting the form into rows of a table).

7. While adding features in, there is an issue with deciding whether to allow a non-js alternative to them. This will require a lot more flexibility and planning.