User Manual

// file directory structure (including only MVC)

```
task_manager/
 + app
  + models
    | tag.rb
    | task.rb
    | threshold.rb

  + views
   + shared
     | _nav.html.erb  // navigation bar
   + tags
     | index.html.erb
   + tasks
     | _form.html.erb // form for create/update operations
     | _list.html.erb // automate screening of tasks by priority level and due date
     | edit.html.erb
     | index.html.erb
     | new.html.erb
     | show.html.erb
   + thresholds                // thresholds contains only one entry
                                        (number of days from now) that
                                        separates due earlier and due later
     | edit.html.erb
     | index.html.erb

  + controller
    | tag.rb
    | task.rb
    | threshold.rb
```

Preamble

This task manager supports CRUD operations of tasks. Tasks are organised according to their priority levels and due dates. User can specify the priority level while creating the task and change the threshold value separating DUE EARLIER tasks and DUE LATER tasks at the settings page.

Tasks and tags are associated via the has_and_belongs_to_many property, where a link table is created to store records of the associated IDs of linked tasks/tags.

User can create tags after tasks are created (when viewing tasks), where the delete/edit options for tasks and the delete operation for tags are found.

Tags are destroyed only when all associated tasks are destroyed. In other circumstances, only records in the link table are destroyed.

Key Takeaways

1. Understood the advantages of the MVC model for organising web content. Realised that such a model might also limit the flexibility in website design (but saves significant amount of time and resources).

2. Implemented CRUD operations for rails.

3. Implemented the more complicated has_and_belongs_to_many association and handled the complications that arose, for example, to destroy a tag only when all associated tasks are destroyed.

4. Understood how to DRY in rails, including the use of partials and creating shared folders between different view components.

5. Integrated bootstrap well with rails application

6. Set up the remote Ubuntu server to serve the rails app

7. Picked up Ruby and the rails API