# Todo-Manager Final-Submission

## User Manual

***Basic Information***

Todo-Manager is built based on Ruby on Rails, with the Rails installation version 5.1.4 and Ruby version 2.3.5 (used to be 2.4.3). Bootstrap, Devise, and Ransack are the three main gems that are used during the development period.

***Use Cases and Functions***

It is an electronic memo pad, which aims to assist people with remembering and managing their tasks in daily life. With Todo-Manager, it will be easier for users to analyse and decide the priority when doing tasks. In order to make it more flexible and user-friendly, the following functions are provided:

*1. the basic CRUD function (for groups and tasks)*

Todo-Manager allows you to **create**, **read**, **update** and **delete** groups as well as its following tasks. For groups, you only need to provide name; while for tasks, text, and group must be provided. (Although Todo-Manager doesn't check the existence of deadline, there is actually no way for you to leave deadline blank.) Moreover, bulk_delete is accessible in Todo-Manager, click on the select boxes in front of task-text in the group-detail or view-all page and then delete, all the tasks that have been selected will be deleted immediately.

NOTE: You MUST have **at least ONE** group before you continue to create tasks. If you delete a group, all of its containing tasks will be deleted as well.

*2. classification of tasks*

While creating or editing tasks, you can select which group is the task in. If you create a task in a group-detail page, the default choice of the group of the new task will be that group by default, but you can change it if you want to assign the task to another group. (e.g. You want to create a task when you are at General group-detail page, the default group of your new task will be General.) The group where the task is assigned will decide in which group-detail page the task will appear. In the meantime, if you prefer viewing all tasks at the same time, the group assigned to the tasks will also be shown in the view-all page so that it will be easier for you to manage overall tasks.

NOTE: Todo-Manager allows one task belongs to **only one** group.

3. *highlight urgent tasks*

You are able to highlight urgent tasks by **selecting the radio button** when creating or editing tasks. As emergency is not an attribute that tasks must have, you are allowed to choose neither option. In this situation, Todo-Manager treats this as choosing NO by default. Highlighting can remind you of urgent tasks.

*4. search*

In order to help you to find out the specific task that you may want to have a look at, Todo-Manger provides a simple search function which is built based on *ransack*. It allows you to search tasks with the keyword. As long as the word appears in the text attribute of one task, that task will turn out to be one of the search results. This can be used as another grouping if you manage to set the text of tasks properly. (e.g. You can find all tutorials as long as you attach a single for that in the text of tasks, even if you divide tasks into groups based on module code.)

NOTE: Groups won't take part in the searching function. Spelling mistakes or too many keywords may not turn out expected results.

*5. navigation bar*

The navigation bar is built with *bootstrap* for Todo-Manager. You can either click on Todo-Manager to get to the homepage (where all groups appearing without their tasks) or click on group names to get to the detail page of that group (which contains all tasks of that group). It should be noted that user based options (sign in/out, register, etc.), as well as search function are also shown in the navigation bar.

*6. user authentication*

User authentication is built based on *devise*. You have to log in in order to get access to other pages except for the welcome page. Moreover, you will only have access to your own created groups and tasks. Therefore, there is no need to worry about your list being messed up by others' lists. Todo-Manager is a private helper for you.

## My Accomplishment

Honestly this is the first time I try to reach something that I have never known of before and I did suffer from that at the beginning. However, I am satisfied with myself that I finally made it. During this period, except for the most basic way to build a ruby application, I learned about some ideas that I can use in the future. In the following paragraphs, I will share my gains as well as my review of Todo-Manager.

### Gains

*Less duplication*

Firstly, it is senseless to keep repeating myself when programming. Rails provides several ways to keep me away from that, such as layout and partial template. Besides, DRY (Don't Repeat Yourself) is also a spirit that it advocates, which shares the same meaning about this programming methodology. I feel that less repeat makes my life easier and my program neater. It brings convenience especially when I need to change the template that is used in many areas, such as navigation bar.

*Use tools*

Secondly, it is important to be able to find appropriate tools to help me, like gem. With the improvement of computer science, a lot of useful tools are provided online. Although some of them may fail to supply the most efficient way to build up a function, we should seed their help based on the situation. In order to save time and manpower, the ability to search for useful tools should be necessary for the industrial production process, as long as the efficiency of the final version is acceptable.

*Consider more situations*

Thirdly, it is beneficial to consider exceptions instead of pretending that users will follow my instruction. When building an application, one of the most important things to consider is user experience. My last version of this application just tells users that they should build a group before creating new tasks, and assume that everyone will follow my instruction. However, when I ask my friends to test for me, there are some of them having ignored my instruction and ran into errors. Therefore I realise that building an application is different from doing labs. When doing labs, I can ignore the unexpected situations and do nothing for that, however, when it comes to building an application, I have to make sure that there are as few as confusing errors that affect my users. I need to make sure I consider as many situations as possible.

## Review of Todo-Manager

*Advantages*

In my opinion, Todo-Manager manages to provide enough functions for users and will be able to handle daily life needs. Its web pages are neat and should be clean enough for users to check their tasks easily.

For source code, I manage to use as less duplication as possible, which means I made full use of before_action, layout as well as partial templates. That will bring some convenience to future update and code reading.

*Disadvantages*

There are several problems with the user model. Users are not properly protected. Change group_id in the URL can easily give you access to others' groups. And email address is not checked properly enough, for example, you can simply type in an invalid email address which has the basic email address format, such as 123@gmail.com. Todo-Manager will let you create this account even if its invalid.

Lastly, I must admit for some parts I used tricks, which means sometimes I am not really sure about why there used to be an error and I just guess where the problem is and find a way to avoid it. For example, when the input of the name of the first group that one user created is empty, there will be an error about no group_id for the link inside navigation bar. I don't really know about why it turns out to be that, but I find that the problem must come from group_id and maybe _navbar mistakes the invalid new group to be one of @groups. Therefore, I check whether group.id exists. However, the problem should only come from the last created group without a name. There should be a more efficient and proper way to deal with this. This is a problem about efficiency. If the number of groups is large, there will be too many extra checks. Although it seems that this doesn't matter much in this situation, for future study, a better way should be provided.