

## CVWO Mid-Assignment Submission

## Use Case &amp; Execution plan

## Creating todo model

In bash, enter the command

```
rails generate model Todo item:string category:string
```

Migrate todo table to database with

```
rake db:migrate
```

Add test data by opening the rails console.

```
rails console
```

Enter command in console.

```
Todo.create(:item => "Do mission 8", :category => "School")
```

## Viewing todo items

Go to localhost:3000.

Todo items are listed in the homepage.

## TODO List



## Execution

Under config/routes.rb, add the following to create restful routing.

```
resources :todos
```

Create Todo controller.

```
rails generate controller todos
```

Under file app/controllers/todos\_controller.rb, assign a todo model instance to the index page.

```
def index
  @todos = Todo.all
end
```

Create index.html.erb under app/views/todos.

Add the following codes to list todo items.

```
<% @todos.each do |todo| %>
  <ul>
    <li>
      <span class="todo_items"><%= todo.item %></span>
      <button><%= link_to "Edit", '#' %></button>
      <button><%= link_to "Delete", '#' %></button>
    </li>
  </ul>
<%end%>
```

## Adding a new todo

To add a todo, click the “+” button.

Enter a todo and the category inside the text fields and click “Create Todo”.

## Create Todo

New todo will be displayed in the homepage.

### Execution

Under app/views/todos/index.html.erb, add the following

```
<button><%= link_to "+", new_todo_path %></button>
```

Under file app/controllers/todos\_controller.rb, assign a new todo model instance to the new page.

```
def new
  @todos = Todo.new
end
```

Create new.html.erb under app/views/todos.

Add the following codes to create a form for entering todo attributes.

```
<%= form_for(@todo) do |f| %>
  Todo:<%= f.text_field :item %><br/>
  Category:<%= f.text_field :category %><br/>
  <%= f.submit %>
<% end %>
```

Under file app/controllers/todos\_controller.rb, create a new todo with parameters submitted from new.html.erb.

```
def create
  @todo = Todo.create(todo_params)
  redirect_to todos_path
end

private


def todo_params
  params.require(:todo).permit(:item, :category)
end
```

### Changing todo item

To edit a todo, click the “Edit” button beside the todo.

Change the item description and click “Update Todo”.

## Edit Todo



Updated todo will reflect on the homepage.

### Execution

Under app/views/todos/index.html.erb, change the path for editing todo items.

```
<span class="todo_items"><%= todo.item%></span>
<button><%= link_to "Edit", 'edit_todo_path(todo.id)' %></button>
<button><%= link_to "Delete", '#' %></button>
```

Under app/controller/todos\_controller.rb, before the “private” label, assign model instance of todo to edit page by filtering the id.

```
def edit
  @todo = Todo.find_by(params[:id])
end
```

Create edit.html.erb under app/views/todos.

Add the following codes to create a form for editing todo attributes.

```
<%= form_for(@todo) do |f| %>
  Todo: <%= f.text_field :item %> <br/>
  Category: <%= f.text_field :category %>
  <%= f.submit %>
<% end %>
```

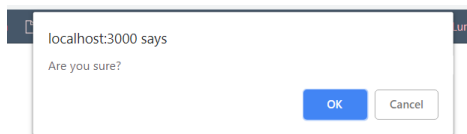
Under file app/controllers/todos\_controller.rb, edit todo with parameters submitted from edit.html.erb.

```
def update
  @todo = Todo.find_by(params[:id])
  @todo.update(todo_params)
  redirect_to todos_path
end
```

### Deleting todo

To delete a todo, click “Delete” button beside the todo.

An alert will be displayed for confirmation.



Press “OK” to confirm deletion.

Homepage will display “\*Delete success”.

### Execution

Under app/views/todos/index.html.erb, change the path for deleting todo items.

```
<span class="todo_items"><%= todo.item%></span>
<button><%= link_to "Edit", `edit_todo_path(todo.id)`%></button>
<button><%= link_to "Delete", todo_path(todo), method: :delete, data: { confirm: 'Are you
sure?' } %></button>
```

Under app/controller/todos\_controller.rb, delete todo item with id given from index.html.erb.

```
def destroy
  @todo = Todo.find(params[:id])
  @todo.destroy
  redirect_to todos_path, notice: "*Delete success"
end
```

### Features intending to add/improve

1. Creation of categories.
2. Show todo lists according to category.
3. Easier use by editing with double click, choosing category with drop down etc.
4. Styling site.

### Problems faced

1. Currently adding category, problem with getting model instances from both ‘Category’ and ‘Todo’ and referencing. (More research to be done)