Sprint Retrospective Report

Date Submitted:

**Team**: Group 7

**Report Created By**: Jacob Haas and Daniel Rivera

**For Sprint**:

**List the goals your team set for this sprint. It is best to name specific features or bug fixes; use the exact issue titles from the GitHub issue tracker**. (*Add more rows if you need*)

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue Title** | **Assigned To** | **Status (Complete, Incomplete, Not Started)** | **Comments (e.g., reasons why feature is incomplete)** |
| View Projects | Daniel | Completed |  |
| View Board for my Project | Daniel | Completed |  |
| failed login (bad credentials) does not generate any feedback | Jacob | Incomplete | Need enhancements |
| edit group/ project should assign member/group using separate box for list of options and list of currently selected items | Jacob | Incomplete | Bug still happening |
| show error when adding group/project on the modal instead of the page behind the modal | Daniel | Completed |  |
| add support for search/filtering large lists | Jacob | Completed |  |
| user should be able to view (but not edit) users, groups, and projects; | Daniel | Completed |  |
| when adding group to project that has a different manager, should require other manager to access project and approve add before fully adding to project | Jacob | Completed |  |
| delete project is failing | Daniel | Completed |  |
| forms should clearly indicate required fields | Daniel | Completed |  |
| should not be able to remove user that is manager for a group or project lead | Daniel | Completed |  |
| Edit Board for Project I manage | Daniel | Completed |  |
| View Board for my Project I am apart of | Daniel | Completed |  |

**For each team member, please provide (via copy/paste) 2-3 examples of evidence they are actively contributing to the development of the team (beyond just coding, etc.). These could include**:

* A git commit message (please include the commit ID and date/time)
* A comment on an issue in the issue tracker (please include the issue name)
* A bug issue they opened (please include the issue name)
* Message(s) from Slack, Discord, etc.

The examples must be substantive, e.g.,

* explaining why a change was made
* providing detail as to how to reproduce a bug
* suggest options to the team for how to solve a particular problem

|  |  |
| --- | --- |
| **Team Member** | **Example Communication(s)** |
| Daniel | **Commit 0b08d7b**  Managers can now edit the boards they manage  3/10/2025 |
| Daniel | **Commit 883d9c1**  Final Kanban Board Stories  3/07/2025 |
| Jacob | **Commit 1b7e61d**  Added Search Functionality  3/10/2025 |
| Jacob | **Commit e8e08e1**  Updated UI adding/editing projects  3/09/2025 |

**List your team’s goals for the next sprint.** (*Add more rows if you need*)

|  |  |  |
| --- | --- | --- |
| **Priority** | **Issue Title** | **Assigned To** |
| 1 | View Task | Jacob |
| 1 | Create Task | Jacob |
| 1 | Update Stage for Task | Daniel |
| 1 | Restrict Column to group | Daniel |
| 1 | edit group/ project should assign member/group using separate box for list of options and list of currently selected items | Daniel |
| 1 | Assign Task | Jacob |

# Report on Generative AI Use

**For every session where you used generative AI, give the following:**

1. **Name(s) of team member(s) who created the prompts**
2. **A copy of your session including both prompts and responses provided in a separate document (one per session) with the name of the document included here**
3. **Purpose of the session (e.g., generate example code for connecting to SQL DB)**
4. **ID of the commit that includes the related source code or documentation changes (state “no changes made” if the session did not lead to any changes)**

**Daniel –**

**Purpose: Continuing more specific implementations of the Kanban Board**

**Commit ID: Commit 0b08d7b**

**To dynamically render the Kanban Board, you can use a Razor View that iterates over the projects and their respective columns and tickets. You’ll want to make use of ViewBag to pass user permissions and restrict actions accordingly. Here’s a structured approach:**

1. **Project Selection – List all user projects so they can switch between them.**
2. **Column Management – Allow adding, renaming, and deleting columns if the user has the required permissions.**
3. **Ticket Display & Drag-and-Drop – Show tickets and allow moving them between columns if permitted.**
4. **Modals for Actions – Use modals to manage tickets and columns.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**This approach ensures that projects and their Kanban boards are displayed dynamically, with permissions properly handled. You should also implement JavaScript functions for drag-and-drop behavior, modal interactions, and AJAX requests for smooth user interaction.**

**Let me know if you need help with specific functionality!**

**ChatGPT:  
To implement the JavaScript logic for your Kanban board, you'll need to use event listeners to handle drag-and-drop interactions, AJAX requests to update the backend, and modal controls for adding, editing, and deleting tickets.**

**Here’s a structured approach:**

1. **Drag-and-Drop Handling**
   * **Attach dragstart events to tickets and columns.**
   * **Use dragover and drop events to allow repositioning.**
   * **Ensure that ticket movements are validated before updating the backend.**
2. **AJAX Integration**
   * **Use fetch to send updates when tickets are moved.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Jacob – For issue:**edit group/ project should assign member/group using separate box for list of options and list of currently selected items  
  
I need help modifying my project and group management UI in my ASP.NET MVC project. When editing a project or a group, I want to show two separate lists: one for available members/groups and one for currently selected members/groups. Users should be able to move selections between these lists. How can I implement this effectively?

**ChatGPT:**

You can achieve this by using two <select> elements, one for available groups/members and one for selected ones. JavaScript functions can handle moving options between these lists.

Here’s an approach:

1. **Create Two List Boxes**
   * One for available groups/members.
   * One for selected groups/members.
2. **Add Buttons to Move Items Between Lists**
   * Use JavaScript to transfer options between <select> elements.
3. **Update the Backend with the Selected Items**
   * Use a hidden input to store selected IDs before submitting the form.

<div class="row">

<div class="col">

<label class="form-label">Available Groups</label>

<select id="availableGroups" class="form-control" multiple>

@foreach (var group in Model.Groups)

{

<option value="@group.Id">@group.Name</option>

}

</select>

</div>

<div class="col-auto d-flex flex-column justify-content-center">

<button type="button" class="btn btn-sm btn-primary mb-2" onclick="moveSelected('availableGroups', 'selectedGroups')">>></button>

<button type="button" class="btn btn-sm btn-primary" onclick="moveSelected('selectedGroups', 'availableGroups')"><<</button>

</div>

<div class="col">

<label class="form-label">Selected Groups</label>

<select id="selectedGroups" class="form-control" multiple name="SelectedGroupIds">

@foreach (var group in Model.Groups.Where(g => Model.SelectedGroupIds.Contains(g.Id)))

{

<option value="@group.Id">@group.Name</option>

}

</select>

</div>

</div>  
  
function moveSelected(fromId, toId) {

var from = document.getElementById(fromId);

var to = document.getElementById(toId);

var selectedOptions = [...from.selectedOptions];

selectedOptions.forEach(option => {

to.appendChild(option);

});

}  
  
  
**Jacob – For issue:**when adding group to project that has a different manager, should require other manager to access project and approve add before fully adding to project  
  
**To enforce this requirement, you can introduce a pending approvals system where group additions require manager confirmation before becoming active.**

**Here’s how you can structure it:**

1. **Track Pending Approvals**
   * **Modify your database to store pending approvals.**
   * **Display pending approvals in the UI.**
2. **Notify the Project Manager for Approval**
   * **Show pending group additions in the project manager’s dashboard.**
   * **Provide an Approve button to confirm the addition.**
3. **Approve the Addition via AJAX**
   * **Once approved, update the database to mark the group as assigned.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**