## Coding standards

- Style coding standard
  - camelCase for variables.
  - lovercase underscored function names.
  - docstrings included in all functions.
- Use of pair programming to develop and validate code
  - For each feature there will be a feature lead who is gonna do the main coding bit for that particular task will the other person keeps a check on the quality and functionality of the code.
  - The roles with rotate with each every new feature and the features would be bigger tasks broken down into smaller ones in order to maintain the pace of development.
- Front end works first on new things that back end needs
  - Once finished coding, notify back end team with a screenshot in the Teams channel.
  - Once approved by the back end team, push to the front end branch.
  - Contact the testing lead.
  - Once approved by the testing lead, he will push it to the dev branch.
  - Back end pulls changes from the dev branch to implement functionality.
- Front end only use REM, no:
  - pixels
  - ems
  - %
- Different styling sheets for every component
- If working on two components within the team, they should have no impact on each other
- If working on two components within the team and you need to change parts of another persons code, notify them

---

## Unified Commenting Format

## BaseTemplatesSection.jsx

Below is an example file that demonstrates the Uniform Commenting style the group complies with

BaseTemplates... ion.txt
26 Oct 2025, 10:11 AM

```jsx
/**
 * BaseTemplatesSection Component
 *
 * This component renders a section displaying base templates for different user types.
 *
 * @component
 * @param {Object} props - Component props
 * @param {string} props.userId - The current user's ID
 * @param {string} props.userType - The type of user ("admin" or "coordinator")
 * @param {Array} props.templates - Array of template objects with id, title, and subject_id
 * @param {Function} props.onBaseTemplateClick - Callback when a base template is clicked
 * @param {Function} props.onCreateFromScratchClick - Callback when "create from scratch" is clicked
 * @returns {JSX.Element} The BaseTemplatesSection component
 */

```

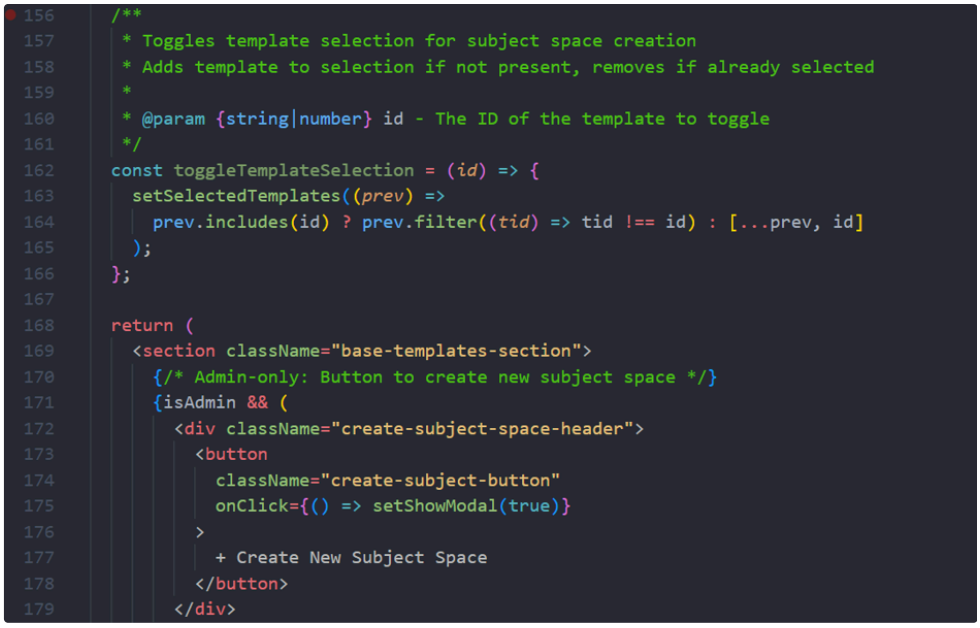
Basic Description and Parameters explained on the top

```jsx
/**
   * Handles creation of a new subject space
   * Validates input fields and makes API call to create subject coordinator account
   * Resets form state and closes modal on success
   */
  const handleCreateSubject = async () => {
    if (!newUsername || !newPassword) {
        showPopup("Please fill in username and password.", "error");

      return;
    }

    try {
      // API call to create subject space
      const res = await fetch(`${HOST}/create_subject_space`, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
          username: newUsername,
          password: newPassword,
          allowed_templates: selectedTemplates,
        }),
      });
```

Function Definitions before each function | in-line comments to explain each chunk of code (.jsx)



```
/**
 * Toggles template selection for subject space creation
 * Adds template to selection if not present, removes if already selected
 *
 * @param {string|number} id - The ID of the template to toggle
 */
const toggleTemplateSelection = (id) => {
  setSelectedTemplates((prev) =>
    prev.includes(id) ? prev.filter((tid) => tid !== id) : [...prev, id]
  );
};

return (
  <section className="base-templates-section">
    {/* Admin-only: Button to create new subject space */}
    {isAdmin && (
      <div className="create-subject-space-header">
        <button
          className="create-subject-button"
          onClick={() => setShowModal(true)}
        >
          + Create New Subject Space
        </button>
      </div>
```

Function Definition 2 (with parameters specified) | in-line comments to explain each chunk of code (.html)