Overview:

The project appears to be a web application built using React. It's a note-taking application with features to create note groups, add notes to those groups, and view the details of each note.

Components:

1. App Component:

- **Purpose**: Acts as the root component of the application.
- **Functionality**:
    - Renders the main layout of the application.
    - Includes the `Page` component, which contains the entire note-taking interface.

2. Page Component:

- **Purpose**: Represents the main layout of the note-taking application.
- **Functionality**:
    - Contains the left and right containers for displaying note groups and note details, respectively.
    - Renders the form for creating new note groups.
    - Manages state related to the visibility of the create button, note titles, selected note group, and mobile view.
- **Subcomponents**:
    - **NoteGroupList**: Displays a list of note groups.
    - **NoteDetails**: Displays the details of the selected note group.
    - **CreateNoteGroupForm**: Provides a form to create new note groups.

3. NoteGroupList Component:

- **Purpose**: Displays a list of note groups.
- **Functionality**:
    - Maps through the `notesTitle` array to render individual note group items.
    - Each note group item represents a clickable element that allows the user to select a note group to view its details.
- **Subcomponents**:
    - **NoteGroupItem**: Represents an individual note group item

4. NoteDetails Component:

- **Purpose**: Displays the details of the selected note group.
- **Functionality**:
    - Includes the list of notes, a text area for adding new notes, and a submit button.
    - Renders each note item with its data, date, and time.
    - Manages state related to the input data for new notes and the selected note group.
- **Subcomponents**: None

5. CreateNoteGroupForm Component:

- **Purpose**: Provides a form for creating new note groups.
- **Functionality**:
    - Renders input fields for entering the group name and selecting a color.
    - Allows users to submit the form to create a new note group.
    - Manages state for the group name and selected color.
- **Subcomponents**: None
    -

State Management:

- **useState** hook is used to manage local component state.
- State variables include:
    - `createButtonVisible`: Controls the visibility of the create button.
    - `notesTitle`: Stores the list of note groups.
    - `notesData`: Stores the data of each note.
    - `selectedGroup`: Represents the index of the selected note group.
    - `isMobileView`: Indicates whether the application is viewed on a mobile device.

Effects:

- **useEffect** hook is used for side effects like fetching data and handling window resize events.
- Fetches and saves `notesTitle` data to localStorage when the component mounts and updates.
- Handles window resize events to adjust the mobile view layout.

Functionality:

- User can create new note groups by providing a name and selecting a color.
- User can select a note group to view its details.

- User can add new notes to the selected note group.
- Notes are stored locally using localStorage.
- Mobile-responsive layout is implemented to adjust the UI based on screen size.

External Libraries:

- **FontAwesome**: Used for icons.

Project Structure:

- Components are organized into separate files within the `Components` directory.
- Styles are managed using CSS files associated with each component.

Code Part:-

CreateNoteGroupForm

1. Purpose:

The `CreateNoteGroupForm` component is responsible for rendering a form that allows users to create a new note group. It provides input fields for entering the group name and selecting a color for the title of the note group.

2. State:

- `groupName`: Manages the value of the group name entered by the user.
- `selectedTitleColor`: Manages the color selected by the user for the title of the note group.

3. Props:

- `createButtonVisible`: Controls the visibility of the form. It's used to conditionally apply a CSS class to show or hide the form.
- `setCreateButtonVisible`: Function to toggle the visibility of the create button. It's called after form submission to hide the create button.
- `addNoteGroup`: Function to add a new note group. It's called upon form submission with the provided group name and selected color.

4. Form Structure:

- The form consists of a header, content, and submit button.
- The header (`form-header`) displays the text "Create New Notes Group".
- The content (`form-content`) contains input fields for the group name and color selection.
- The color selection options are displayed as circles (`color-circle`) with different background colors. Clicking on a color circle updates the selectedTitleColor state.
- The submit button (`form-submit-button`) triggers the form submission. Upon submission, the `handleSubmit` function is called.

5. Event Handlers:

- `handleSubmit`: Handles form submission. It prevents the default form submission behavior, calls the `addNoteGroup` function with the provided group name and selected color, and clears the `groupName` state.
- Clicking on a color circle updates the `selectedTitleColor` state.

6. CSS Classes:

- The CSS classes are used to style the form and its elements, such as setting margins, padding, colors, and borders.

7. Conditional Rendering:

- The form is conditionally rendered based on the value of `createButtonVisible` prop. If `createButtonVisible` is true, the form is visible; otherwise, it's hidden.

NOTES GROUP LIST

1. Purpose:

The `NoteGroupList` component is responsible for rendering a list of note groups. It displays each note group as an item with its title and initials.

2. Props:

- `notesTitle`: An array containing objects representing note groups. Each object contains properties like `id`, `groupName`, and `color`.

- `handleGroupSelection`: A function to handle the selection of a note group. It's called when a note group item is clicked.
- `selectedGroup`: The index of the currently selected note group.
- `isMobileView`: A boolean flag indicating whether the view is in mobile mode or not.

3. Rendering:

- The component returns a `div` element with a class name of `notesTitle-list`.
- Inside this `div`, it maps through the `notesTitle` array and renders each note group item.

4. Note Group Item:

- Each note group item is represented by a `div` element with a class name of `note-item`.
- The background color of each item is determined based on the `isMobileView` and `selectedGroup` props. If it's in mobile view or the item is selected, it sets the background color to `#f7ecdc`, otherwise, it's white.
- Clicking on a note group item triggers the `handleGroupSelection` function with the index of the clicked item.

5. Note Group Title:

- The title of each note group is displayed inside a `div` element with a class name of `note-name`.
- The title is obtained from the `groupName` property of the note group object.

6. Note Group Initials:

- The component utilizes the `getGroupInitials` function to extract initials from the group name.
- If the group name contains more than one word, it takes the first letter of each word to form the initials. Otherwise, it takes the first two letters of the group name.
- These initials are displayed inside a circular div with a class name of `note-color-circle`.

7. CSS:

- The component relies on external CSS styles defined in `styles.css` to handle the layout, colors, and borders of the note group items.

NOTES DETAIL

Purpose:

The `NoteDetails` component is responsible for displaying the details of a selected note group. It shows the notes associated with the selected group, along with options to add new notes and navigate back to the note group list.

2. Props:

- `selectedGroup`: The index of the currently selected note group.
- `notesTitle`: An array containing objects representing note groups. Each object contains properties like `id`, `groupName`, `color`, and `notesData`.
- `notesData`: The data of the notes within the selected group.
- `setNotesData`: A function to update the state of the notes data.
- `handleNoteData`: A function to handle the submission of new note data.
- `handleBackButton`: A function to handle navigating back to the note group list.

3. Rendering:

- The component returns a fragment containing several sections: header, content, and footer.
- Header Section:
    - It includes a back button, note group color circle, and note group name.
    - The back button triggers the `handleBackButton` function.
    - The color circle represents the color of the note group, and the initials are derived from the `groupName`.
- Content Section:
    - It displays the notes associated with the selected group.
    - Each note is rendered within a `div` with class `right-content-data`.
    - Note information such as time and date is displayed using `formatTime` and `formatDate` functions.
- Footer Section:
    - It consists of a text area for entering new notes and a right arrow icon for submitting notes.
    - The text area's value is controlled by the `notesData` state.
    - Pressing Enter without Shift triggers the `handleNoteData` function to submit the note.

4. Functions:

- `formatDate(dateTime)`: Formats the given date and returns a string in the format "day month year".
- `formatTime(dateTime)`: Formats the given time and returns a string in the format "hour
  ".
- `getGroupInitials(groupName)`: Extracts the initials from the group name and returns a string with the first two letters of each word capitalized.
- `handleNoteData()`: Handles the submission of new note data.
- `handleBackButton()`: Handles navigating back to the note group list.

5. Images:

- The component imports left and right arrow icons from image files.
- These icons are used for the back button and submitting new notes.