TaskMaster Application Documentation

This document provides a technical overview of the TaskMaster application, detailing its architecture, components, and how different parts of the code interact.

1. Application Overview

TaskMaster is a single-page application (SPA) built with HTML, CSS, and vanilla JavaScript. It is designed to be a personal task management tool. The application is divided into two main parts: an authentication page (`index.html`) and the main application page (`taskmaster-app.html`). Data is persisted in the browser's `localStorage`.

2. Core Features

Authentication (`index.html`)

- **User Login**: Users can log in with an existing email and password. The application checks for a match in the `localStorage` database of users.
- **User Registration**: New users can create an account by providing a username, email, and password. The new user data is stored in `localStorage`.
- **Password Visibility Toggle**: The login and registration forms include an icon to toggle the visibility of the password field, improving user experience.

Main Application (`taskmaster-app.html`)

- **Task Management**: Users can add, edit, complete, and delete tasks. Each task has a title and a description.
- **Task Filtering**: The application supports filtering tasks by their status:
  - `Active`: Tasks that are not yet completed.
  - `Completed`: Tasks that have been marked as finished.
  - `Deleted`: Tasks that have been soft-deleted.
- **User Profile**: A dedicated profile section displays user information, including their name, email, join date, and task statistics (tasks created and completed). Users can choose from a list of predefined avatars.
- **Settings**: The settings page allows for application-wide customization:
  - **Theme Selection**: Users can choose between multiple themes (Sapphire, Emerald, Amethyst, Crimson) and a dark mode.
  - **Font Size Adjustment**: A slider control allows users to change the base font size for the entire application.
  - **Default View**: Users can set their preferred default task filter view (e.g., Active, Completed).
  - **Data Management**: Features to export tasks to a JSON file and import tasks from a JSON file.

3. Code Structure and Logic

HTML Files

- index.html: Contains the HTML for the login and registration forms. It uses JavaScript to toggle between these two sections and handle form submissions.
- taskmaster-app.html: This is the main application interface. It's composed of several <section> elements, each representing a different view (tasks, profile, settings). The display: none style is used to show/hide sections.

CSS Files

- styles.css: This file contains the base styles for the entire application, including typography, colors, and the layout for the authentication pages.
- taskmaster-app.css: This file contains all the specific styles for the main application, including layouts for the task list, settings, and profile pages. It also defines CSS custom properties (--primary, --dark, etc.) for each of the available color themes and the dark mode.

JavaScript Logic (found within HTML files)

- **Local Storage**: The application heavily relies on localStorage for data persistence. User data, tasks, and settings are all saved and retrieved from localStorage.
  - taskmaster_users: Stores an array of all registered user objects.
  - taskmaster_current_user: Stores the currently logged-in user's object.
  - taskmaster_tasks: Stores the array of tasks for the current user.
  - taskmaster_settings_theme, taskmaster_settings_font_size, etc.: Store individual user settings.
- **DOM Manipulation**: The JavaScript code directly manipulates the Document Object Model (DOM) to dynamically render tasks, update profile information, and handle UI state changes (e.g., showing/hiding sections, applying themes).
- **Event Listeners**: The code uses numerous event listeners ('click', 'change', 'submit', 'DOMContentLoaded') to respond to user actions and manage application flow.
- **Data Handling**: Functions like loadTasks(), saveTasks(), renderTasks(), addTask(), updateTask(), and deleteTask() encapsulate the logic for managing task data.

4. Key Functions and Components

- showSection(sectionId): A function that hides all content sections and shows only the one with the specified ID.
- loadTasks(): Retrieves tasks from localStorage and calls renderTasks().
- renderTasks(tasksToRender): Generates the HTML for the task list and inserts it into the DOM.
- saveTasks(): Serializes the tasks and deletedTasks arrays to JSON and saves them to localStorage.
- loadProfile(): Retrieves the current user's data from localStorage and populates the profile section.
- handleThemeChange(): Updates the <body> class to apply a new theme.
- setFontSize(size): Dynamically changes the font size of the <body> element.
- exportData(): Creates a JSON object of tasks and deleted tasks and allows the user to download it as a file.
- importData(): Reads a JSON file selected by the user and loads the data into the application.