AGILE BOARD

Kanban Task Management App

PRESENTED BY QAIYLAH CONWAY

IMPORTING HELPER FUNCTIONS FROM UTILS & INITIALDATA

All the initialdata has been imported from the initialData.js file

The getTasks, createNewTask, patchTask, putTask and deleteTask have been imported from the taskFunctions.sj file

The function checks if the value associated with the key 'tasks' in the browser's localStorage is falsy (not set or empty). If the 'tasks' value is falsy, it initializes it with a stringified version of an object called initialData and sets the 'showSideBar' key to 'true'. If the 'tasks' value already exists, it logs a message to the browser console indicating that data is already present in localStorage.

index.html | index.js | initialData.js

index.js > deleingTheTaskfomDom

```
// TASK: import helper functions from utils - solved
import {
  getTasks,
  createNewTask,
  patchTask,
  putTask,
  deleteTask} from "/utils/taskFunctions.js"

// TASK: import initialData - solved
import {initialData} from "/initialData.js";

/*********************************************************************************************************************
 * FIX BUGS!!!
 * ******************************************************************************************************************

// localStorage.clear() // clear storage

// Function checks if local storage already has data, if not it loads initialData to localStorage
function initializeData() {
  if (!localStorage.getItem('tasks')) {
    localStorage.setItem('tasks', JSON.stringify(initialData));
    localStorage.setItem('showSideBar', 'true')
  } else {
    console.log('Data already exists in localStorage');
  }
}
```

GETTING ELEMENTS FROM THE DOM

The code snippet defines an object called `elements` that holds references to various HTML elements selected by their unique identifiers or classes. It also declares a variable `activeBoard` and sets it to an empty string.
document.querySelector(...): This method returns the first element within the document that matches the specified selector. For example, line 30, document.querySelector("#header-board-name") selects the element with the ID header-board-name.

BUGS FIXED

The only bug I've fixed in the fetchAndDisplayBoardAndTasks function was changing the semicolon to a colon in line 50, for it is the correct character to use in order to get the code working and activate the board.

index.html | index.js | initialData.js

index.js > ...

```
27
28  // TASK: Get elements from the DOM - solved
29  const elements = {
30    headerBoardName: document.querySelector("#header-board-name"),
31    columnDivs: document.querySelectorAll(".column-div"),
32    editTaskModal: document.querySelector(".edit-task-modal-window"),
33    filterDiv: document.querySelector("#filterDiv"),
34    hideSideBarBtn: document.querySelector("#hide-side-bar-btn"),
35    showSideBarBtn: document.querySelector("#show-side-bar-btn"),
36    themeSwitch: document.querySelector("#switch"),
37    createNewTaskBtn: document.querySelector("#add-new-task-btn"),
38    modalWindow: document.querySelector(".modal-window")
39  };
40  let activeBoard = ""
41
42  // Extracts unique board names from tasks
43  // TASK: FIX BUGS
44  function fetchAndDisplayBoardsAndTasks() {
45    const tasks = getTasks();
46    const boards = [...new Set(tasks.map(task => task.board).filter(Boolean))];
47    displayBoards(boards);
48    if (boards.length > 0) {
49      const localStorageBoard = JSON.parse(localStorage.getItem("activeBoard"))
50      activeBoard = localStorageBoard ? localStorageBoard :  boards[0];
51      elements.headerBoardName.textContent = activeBoard
52      styleActiveBoard(activeBoard)
53      refreshTasksUI();
54
55    }
56  }
```

DIFFERENT BOARDS IN THE DOM

The displayBoards function primarily generates buttons for individual boards, manages click events, updates UI elements, and oversees the active board state.

BUGS FIXED

Added an event listener method, enclosed the 'click' parameter in quotation marks, and incorporated an arrow function.

JS index.js > ...

```
57
58  // Creates different boards in the DOM - solved
59  // TASK: Fix Bugs
60  function displayBoards(boards) {
61    const boardsContainer = document.getElementById("boards-nav-links-div");
62    boardsContainer.innerHTML = ''; // Clears the container
63    boards.forEach(board => {
64      const boardElement = document.createElement("button");
65      boardElement.textContent = board;
66      boardElement.classList.add("board-btn");
67      boardElement.addEventListener('click', () => {  // added eventListener, arrow function
68        elements.headerBoardName.textContent = board;
69        filterAndDisplayTasksByBoard(board);
70        activeBoard = board //assigns active board
71        localStorage.setItem("activeBoard", JSON.stringify(activeBoard))
72        styleActiveBoard(activeBoard)
73      });
74      boardsContainer.appendChild(boardElement);
75    });
76
77  }
78
79  // Filters tasks corresponding to the board name and displays them on the DOM.
80  // TASK: Fix Bugs
81  function filterAndDisplayTasksByBoard(boardName) {
82    const tasks = getTasks(); // Fetch tasks from a simulated local storage function
83    const filteredTasks = tasks.filter(task => task.board === boardName);  // added strictly equal to
84
```

ADDING A filterAndDisplay TasksByBoard Function

BUGS FIXED

Changed all equal (=) signs to strictly equal (===) signs instead. As well as a setAttribute to set the value of the taskContainer attrubute. I've also added an eventListener and quotation marks for the 'click' parameter.

For the styleActiveBoard function I've added classList to manipulate the lists of classes for an HTML element to both the add (btn.classList.add) and remove (btn.classList.remove) buttons.

```
81  function filterAndDisplayTasksByBoard(boardName) {
82    const tasks = getTasks(); // Fetch tasks from a simulated local storage function
83    const filteredTasks = tasks.filter(task => task.board === boardName);  // added strictly equal to
84
85    // Ensure the column titles are set outside of this function or correctly initialized before this function runs
86
87    elements.columnDivs.forEach(column => {
88      const status = column.getAttribute("data-status");
89      // Reset column content while preserving the column title
90      column.innerHTML = `<div class="column-head-div">
91                            <span class="dot" id="${status}-dot"></span>
92                            <h4 class="columnHeader">${status.toUpperCase()}</h4>
93                          </div>`;
94
95      const tasksContainer = document.createElement("div");
96      tasksContainer.setAttribute("class", "tasks-container")
97      column.appendChild(tasksContainer);
98
99      filteredTasks.filter(task => task.status === status).forEach(task => {  // added strictly equal to
100       const taskElement = document.createElement("div");
101       taskElement.classList.add("task-div");
102       taskElement.textContent = task.title;
103       taskElement.setAttribute('data-task-id', task.id);
104
105       // Listen for a click event on each task and open a modal - solved
106       taskElement.addEventListener('click', () => {   // added evenListener
107         openEditTaskModal(task);
108       });
109
110       tasksContainer.appendChild(taskElement);
111     });
```

```
116 function refreshTasksUI() {
117   filterAndDisplayTasksByBoard(activeBoard);
118 }
119
120 // Styles the active board by adding an active class
121 // TASK: Fix Bugs - solved
122 function styleActiveBoard(boardName) {
123   document.querySelectorAll('.board-btn').forEach(btn => {
124
125     if(btn.textContent === boardName) {
126       btn.classList.add('active')   // added classList
127     }
128     else {
129       btn.classList.remove('active');  // added classList
130     }
```

ADDING AN addTaskUI Function

BUGS FIXED

Added refreshTasksUI(); for Visual Feedback: After certain actions like creating, editing, or completing a task, calling this function would provide immediate visual feedback to the user by updating the task list.

initialData.push(task);
localStorage.setItem('tasks',JSON.stringify(initialData))
This code is to update the list of tasks stored in localStorage by adding a new task and then saving the updated list back to localStorage

index.html | index.js | initialData.js

index.js > ...

```
function addTaskToUI(task) {
  const column = document.querySelector(`.column-div[data-status="${task.status}"]`);
  if (!column) {
    console.error(`Column not found for status: ${task.status}`);
    return;
  }

  let tasksContainer = column.querySelector('.tasks-container');

  if (!tasksContainer) {
    console.warn(`Tasks container not found for status: ${task.status}, creating one.`);
    tasksContainer = document.createElement('div');
    tasksContainer.className = 'tasks-container';
    column.appendChild(tasksContainer);
  }

  const taskElement = document.createElement('div');
  taskElement.classList.add('task-div');
  taskElement.textContent = task.title; // Modify as needed
  taskElement.setAttribute('data-task-id', task.id);


  tasksContainer.appendChild(taskElement);
  refreshTasksUI();  // added


initialData.push(task); // added
localStorage.setItem('tasks',JSON.stringify(initialData)) // added

}
```

ADDING A setupEventListeners Function: PART 1

The function setupEventListeners is primarily dedicated to configuring event listeners, as implied by its name. These event listeners have been attached to various objects, including the cancelEditBtn, cancelTaskBtn, filterDiv, as well as the hide and showSideBarBtn.

BUGS FIXED

There was three lines of code which were missing the eventListener in this section of code. They were in line 171, and in the ‘show sidebar event listener section, line 189 and 190.

index.html | index.js | initialData.js

index.js > ...

```
167
168   function setupEventListeners() {
169     // Cancel editing task event listener - solved
170     const cancelEditBtn = document.getElementById('cancel-edit-btn');
171     cancelEditBtn.addEventListener('click', () => { // added eventListener
172       toggleModal(false, elements.editTaskModal)
173     });
174
175     // Cancel adding new task event listener
176     const cancelAddTaskBtn = document.getElementById('cancel-add-task-btn');
177     cancelAddTaskBtn.addEventListener('click', () => {
178       toggleModal(false);
179       elements.filterDiv.style.display = 'none'; // Also hide the filter overlay
180     });
181
182     // Clicking outside the modal to close it
183     elements.filterDiv.addEventListener('click', () => {
184       toggleModal(false);
185       elements.filterDiv.style.display = 'none'; // Also hide the filter overlay
186     });
187
188     // Show sidebar event listener
189     elements.hideSideBarBtn.addEventListener('click', () => toggleSidebar(false)); // added eventListener
190     elements.showSideBarBtn.addEventListener('click', () => toggleSidebar(true));  // added eventListener
191
192     // Theme switch event listener
193     elements.themeSwitch.addEventListener('change', toggleTheme);
```

ADDING A setupEvent Listeners Function: PART 2

The other three objects with the eventListener in this snippet of code would be the themeSwitch, createNewTaskBtn, modalWindow.

BUGS FIXED

The sole issue in this code was traced back to a character error on line 220. The adjustment made involved replacing the arrow function (=>) with a colon (:).

index.html | index.js | initialData.js

index.js > ...

```
168   function setupEventListeners() {
186     });
187
188     // Show sidebar event listener
189     elements.hideSideBarBtn.addEventListener('click', () => toggleSidebar(false)); // added eventListener
190     elements.showSideBarBtn.addEventListener('click', () => toggleSidebar(true));  // added eventListener
191
192     // Theme switch event listener
193     elements.themeSwitch.addEventListener('change', toggleTheme);
194
195     // Show Add New Task Modal event listener
196     elements.createNewTaskBtn.addEventListener('click', () => {
197       toggleModal(true);
198       elements.filterDiv.style.display = 'block'; // Also show the filter overlay
199     });
200
201     // Add new task form submission event listener
202     elements.modalWindow.addEventListener('submit',  (event) => {
203       addTask(event)
204     });
205   }
206
207   // Toggles tasks modal
208   // Task: Fix bugs
209   function toggleModal(show, modal = elements.modalWindow) {
210     modal.style.display = show ? 'block' : 'none';   // changed => to :
211   }
212
213   /*************************************************************************************************************
214    * COMPLETE FUNCTION CODE
215    * ***********************************************************************************************************
```

ADDING AN addTask Function

In this const task object I've added the properties: title, descriptive and status, and assigned each of the values of the input fields with a matching ID.

The board property (board: activeBoard) of the task object is assigned the value of the activeBoard variable. This variable likely holds information about the currently active board or context.

For the toggleSidebar function, the codes means that if truthy it's set to block, if falsy sets to none.

And in the toggleTheme function, from 260 to 264, it means that the togggle will updates to light-theme if strictly true, but if no light then remains in dark-theme.

index.html | index.js | initialData.js

index.js > ...

```
217  function addTask(event) {
218    event.preventDefault();
219
220    //Assign user input to the task object - solved
221      const task = {
222        title: document.getElementById("title-input").value,
223        description: document.getElementById("desc-input").value,
224        status : document.getElementById("select-status").value,
225        board : activeBoard
226      };
227
228      const newTask = createNewTask(task);
229      if (newTask) {
230        addTaskToUI(newTask);
231        toggleModal(false);
232        elements.filterDiv.style.display = 'none'; // Also hide the filter overlay
233        event.target.reset();
234        refreshTasksUI();
235      }
236  }
237
238  // Sidebar - solved
239  function toggleSidebar(show) {
240    const sidebar = document.querySelector(".side-bar")
241    sidebar.style.display = show ? 'block' : 'none';
242    elements.showSideBarBtn.style.display = show ? 'none' : 'block';
243  }
244
```

```
244
245  // Theme - solved
246  function toggleTheme() {
247    const logo = document.getElementById("logo")
248
249    if(document.body.classList.toggle('light-theme') === true) {
250      logo.setAttribute('src', "./assets/logo-light.svg")
251    } else {
252      logo.setAttribute('src', "./assets/logo-dark.svg")
253    }
254  }
```

ADDING AN openEditTaskModal Function

Initialization of Modal Inputs:
The function begins by selecting three elements from the DOM: inputs for the task's title, description, and a select element for the task's status. These are stored in titleInput, descInput, and statusSelect.
It then sets the value of these elements to match the properties of the task object passed to the function (task.title, task.description, task.status)

Event Listeners for Buttons:
Two buttons are selected from the DOM: the "Save Changes" button and the "Delete Task" button, stored in saveTaskChangesBtn and deleteTaskBtn. An event listener is added to saveTaskChangesBtn that, when clicked, calls the saveTaskChanges function with the task's ID as an argument. Another event listener is added to deleteTaskBtn that, when clicked, calls the deleteTask function with the task's ID and then closes the edit task modal.

Modal Visibility Toggle:
Finally, the toggleModal function is called with true and elements.editTaskModal as arguments to display the edit task modal.

index.html | index.js

index.js > openEditTaskModal

```
269  function openEditTaskModal(task) {
270    // Set task details in modal inputs - solved
271
272    const titleInput = document.getElementById("edit-task-title-input");
273    const descInput = document.getElementById("edit-task-desc-input");
274    const statusSelect = document.getElementById("edit-select-status");
275
276    // Get button elements from the task modal - solved
277
278    titleInput.value = task.title;
279    descInput.value = task.description;
280    statusSelect.value = task.status;
281
282    // Call saveTaskChanges upon click of Save Changes button - solved
283
284    const saveTaskChangesBtn = document.getElementById("save-task-changes-btn");
285    const deleteTaskBtn = document.getElementById("delete-task-btn");
286
287    saveTaskChangesBtn.addEventListener("click", () => {
288      saveTaskChanges(task.id);
289    });
290
291    // Delete task using a helper function and close the task modal - solved
292
293    deleteTaskBtn.addEventListener("click", () => {
294      deleteTask(task.id);
295      toggleModal(false, elements.editTaskModal);
296    });
297
298    toggleModal(true, elements.editTaskModal); // Show the edit task modal
299    refreshTasksUI();
300  }
```

ADDING A saveTaskChanges Function

The saveTaskChanges function collects user input for updating a task, creates an object with the updated task details, triggers the task update (using a helper function), closes the modal, and refreshes the UI to reflect the changes

The helper function, whixh is the patchTask, handles the actual update of the task, either in a local data store or on a server.

refreshTasksUI(); code calls to update the user interface

index.html | index.js

index.js > openEditTaskModal

```
301
302   //  saveTaskChanges function - solved
303   function saveTaskChanges(taskId) {
304     // Get new user inputs - solved
305     const titleInput = document.getElementById("edit-task-title-input");
306     const descInput = document.getElementById("edit-task-desc-input");
307     const statusSelect = document.getElementById("edit-select-status");
308
309     const newTitle = titleInput.value;
310     const newDescription = descInput.value;
311     const newStatus = statusSelect.value;
312
313     // Create an object with the updated task details - solved
314     const updateTask ={
315     id: taskId,
316     title: titleInput.value,
317     description: descInput.value,
318     status: statusSelect.value,
319
320   };
321     // Update task using a helper function - solved
322     patchTask(taskId, updateTask);
323
324     // Close the modal and refresh the UI to reflect the changes - solved
325     toggleModal(false, elements.editTaskModal);
326
327     refreshTasksUI();
328   }
329
330   /*****************************************************************************
```

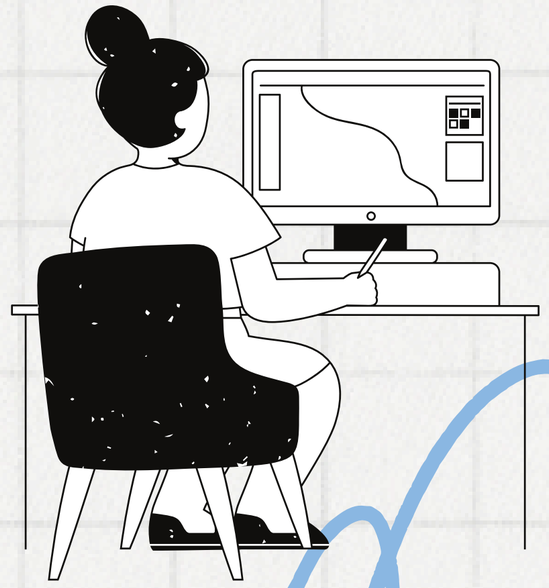
DOCUMENT EVENTLISTENER & INIT FUNCTION

The provided code initializes a web page by setting up data, attaching event listeners, adjusting UI elements based on stored preferences, and displaying initial content

index.html | index.js | initialData.js

index.js > ...

```
295  /*************************************************************************************************
296
297  document.addEventListener('DOMContentLoaded', function() {
298      init(); // init is called after the DOM is fully loaded
299  });
300
301  function init() {
302      initializeData();
303      setupEventListeners();
304      const showSidebar = localStorage.getItem('showSideBar') === 'true';
305      toggleSidebar(showSidebar);
306      const isLightTheme = localStorage.getItem('light-theme') === 'enabled';
307      document.body.classList.toggle('light-theme', isLightTheme);
308      fetchAndDisplayBoardsAndTasks(); // Initial display of boards and tasks
309  }
310
311
```



Thank you very much!

PRESENTED BY QAIYLAH CONWAY