Below is the **collection** of **Sprint Backlogs** for the ToDo List web application project in the Scrum context. These reflect **actual recorded sprints** that the team completed sequentially, with **dates, goals, tasks, estimates, and statuses** showing how the project evolved over time. The structure below demonstrates a clear record of what was planned and how it was carried out.

Contents

[Additional Notes for the Next Sprints 2](#_Toc187927243)

[**Proposed Sprint 2: Important Tasks & Archiving** 2](#_Toc187927244)

[**Proposed Sprint 3: Filtering / Sorting & UI/UX Improvements** 2](#_Toc187927245)

[**Proposed Sprint 4: Advanced Features (Share / Sync or Mobile Integration)** 3](#_Toc187927246)

[**Flexibility & Adaptation** 3](#_Toc187927247)

[**In Summary** 4](#_Toc187927248)

[Sprint 1: Basic Functionality & Data Persistence 5](#_Toc187927249)

[**Selected Product Backlog Items (PBIs)** 5](#_Toc187927250)

[**Sprint Backlog & Tasks** 5](#_Toc187927251)

[**Additional Activities** 6](#_Toc187927252)

[**Sprint 1 Review** 6](#_Toc187927253)

[Sprint 2: Important Tasks & Archive 7](#_Toc187927254)

[**Selected Product Backlog Items (PBIs)** 7](#_Toc187927255)

[**Sprint Backlog & Tasks** 7](#_Toc187927256)

[**Additional Activities** 8](#_Toc187927257)

[**Sprint 2 Review** 8](#_Toc187927258)

[Sprint 3: UI/UX Improvements & Filtering 9](#_Toc187927259)

[**Selected Product Backlog Items (PBIs)** 9](#_Toc187927260)

[**Sprint Backlog & Tasks** 9](#_Toc187927261)

[**Additional Activities** 10](#_Toc187927262)

[**Sprint 3 Review** 10](#_Toc187927263)

[Sprint 4: Accessibility & Prep for Sync 11](#_Toc187927264)

[**Selected Product Backlog Items (PBIs)** 11](#_Toc187927265)

[**Sprint Backlog & Tasks** 11](#_Toc187927266)

[**Additional Activities** 12](#_Toc187927267)

[**Sprint 4 Review** 12](#_Toc187927268)

[Final Notes 13](#_Toc187927269)

# Additional Notes for the Next Sprints

**Proposed Sprint 2: Important Tasks & Archiving**

* **Sprint Goal**:  
  *Enhance task management by distinguishing important tasks and introducing an archive for completed tasks.*
* **Targeted User Stories**:
  1. **Important Tasks (Epic)** – *User Story #3*
     + Create a dedicated list/section for high-priority tasks.
     + Include the ability to add, toggle, and delete important tasks.
     + Integrate data persistence for these tasks in localStorage.
  2. **Archive / History (Epic)** – *User Story #4*
     + Provide an “Archive” feature that moves completed tasks from the main list to a separate history section.
     + Preserve archived tasks for future reference without cluttering the main to-do list.
* **High-Level Acceptance Criteria**:
  1. A separate input/button for “Important Tasks.”
  2. A new “Archive Completed” button, transferring completed tasks to a “History” section.
  3. Both important and archived tasks stored and retrieved from localStorage.
* **Why This Sprint?**
  1. These features **add immediate value**, allowing users to differentiate urgent tasks and keep their main list clear.
  2. Builds upon the **foundation** of basic task management from Sprint 1.

**Proposed Sprint 3: Filtering / Sorting & UI/UX Improvements**

* **Sprint Goal**:  
  *Introduce better organization methods for tasks and refine the user interface for a smoother experience.*
* **Targeted User Stories**:
  1. **UI/UX Improvements** – *User Story #5*
     + Enhance responsiveness and visual design.
     + Ensure buttons, input fields, and layout work well across various screen sizes.
  2. **Filtering / Sorting Tasks** – *User Story #6*
     + Provide options to filter tasks by completed/incomplete status.
     + Allow sorting tasks by name, date, or other criteria.
* **High-Level Acceptance Criteria**:
  1. A polished UI with improved CSS or a simple front-end framework.
  2. Filter controls (e.g., dropdowns or buttons) for completed vs. incomplete tasks.
  3. Sort functionality (alphabetical or by creation date).
  4. Clear, mobile-friendly layout with proper spacing, font sizing, etc.
* **Why This Sprint?**
  1. Improves **usability** and **aesthetics**, encouraging better user engagement.
  2. Filtering and sorting become critical as the number of tasks grows, making the app more scalable for real-world usage.

**Proposed Sprint 4: Advanced Features (Share / Sync or Mobile Integration)**

* **Sprint Goal**:  
  *Begin implementing or laying the groundwork for advanced capabilities, such as multi-device sharing/sync or mobile integration.*
* **Potential Targeted User Stories** (depending on priority):
  1. **Share / Sync Feature (Epic)** – *User Story #7*
     + Introduce a backend or cloud sync mechanism to keep tasks consistent across multiple devices.
     + Possibly implement user-based or link-based sharing for collaboration.
  2. **Mobile App Integration (Epic)** – *User Story #8*
     + Investigate or develop a mobile app (native or hybrid) that syncs with the web version.
     + Use the same data source or a new backend service to maintain real-time consistency.
* **High-Level Acceptance Criteria** (if Share/Sync is chosen):
  1. Basic authentication or user identification to handle multiple user sessions.
  2. Data synchronization method (websockets, periodic polling, or third-party service).
  3. Ability to see updated tasks across different browser sessions or devices.
* **High-Level Acceptance Criteria** (if Mobile Integration is chosen):
  1. A functional prototype or proof-of-concept mobile application using the same data layer.
  2. Responsive design or cross-platform framework (e.g., React Native, Flutter, or a PWA approach).
* **Why This Sprint?**
  1. Represents a **major expansion** of functionality, transforming the app from a single-device tool into a collaborative or multi-platform solution.
  2. By this stage, the product has enough stability and user-facing features to support more advanced use cases.

**Flexibility & Adaptation**

* **Product Owner** and **Scrum Team** will refine priorities before each sprint, choosing which stories to tackle based on the **current needs**, **business value**, and **team capacity**.
* This outline is **not** set in stone; user feedback or newly discovered constraints may cause the team to adjust scope, swap backlog items, or reprioritize the order of epics and stories.

**In Summary**

1. **Sprint 2**: Expand capabilities with “Important Tasks” and “Archive / History.”
2. **Sprint 3**: Improve user **experience** with filtering/sorting and a more polished interface.
3. **Sprint 4**: Tackle **advanced features**—either share/sync for multi-device usage or mobile app integration.

Following this **continuous roadmap**, the team ensures a steady evolution of the to-do application, delivering incremental improvements that align with both **user needs** and **business goals**—all while preserving the agility to adapt if circumstances or user feedback prompt a change in priorities.

# Sprint 1: Basic Functionality & Data Persistence

* **Dates**: April 1 – April 14 (2-week sprint)
* **Sprint Goal**:  
  “Implement core task management and data persistence, ensuring tasks remain saved after page refresh.”

**Selected Product Backlog Items (PBIs)**

1. **Basic Task Management** (5 SP)
   * *As a user, I want to add, mark complete, and delete tasks so that I can manage my daily to-do list effectively.*
2. **Data Persistence** (3 SP)
   * *As a user, I want my tasks to remain saved after I refresh or close the browser, so I never lose track of my tasks.*

**Total**: 8 SP

**Sprint Backlog & Tasks**

**PBI 1: Basic Task Management (5 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 1.1: Add Task Input + Button (UI) | Create a text input and “Add Task” button in index.html; style in styles.css. | 1 SP | Done |
| 1.2: addTask() Function | Implement JavaScript to read user input, validate it, and add a new task to taskList. | 1 SP | Done |
| 1.3: toggleTask() | Allow toggling a task’s completed property by clicking on it or a checkbox. | 1 SP | Done |
| 1.4: deleteTask() | Remove tasks from taskList when clicking the Delete button. | 1 SP | Done |
| 1.5: renderTasks() Function | Render the updated task list in the UI whenever tasks are added, toggled, or deleted. | 1 SP | Done |

**Total**: 5 SP → **All Completed**

**PBI 2: Data Persistence (3 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 2.1: Plan Local Storage (JSON) | Decide on how to store tasks in localStorage. Identify keys/JSON structure. | 1 SP | Done |
| 2.2: saveTasksToLocalStorage() | Serialize tasks (and other arrays if needed) into JSON and store them in localStorage whenever changes occur. | 1 SP | Done |
| 2.3: retrieveTasksFromLocalStorage() | On page load, parse JSON from localStorage and populate the taskList. | 1 SP | Done |

**Total**: 3 SP → **All Completed**

**Additional Activities**

|  |  |  |
| --- | --- | --- |
| Activity | Description | Status |
| Integration / Refactoring | Ensure addTask(), toggleTask(), deleteTask(), and renderTasks() work well together. | Done |
| Peer Code Review | Another developer reviews code, checks logic, and suggests naming improvements. | Done |
| Documentation | Write a short README.md describing setup steps, plus inline JS doc comments. | Done |

**Sprint 1 Review**

* **Outcome**:
  + Users can add, mark complete, delete, and **retain** tasks across sessions.
  + The localStorage feature is functional, meeting acceptance criteria for PBIs #1 and #2.
* **Velocity**: 8 SP completed.
* **Increment**: A basic, working ToDo List app with persistent data.

# Sprint 2: Important Tasks & Archive

* **Dates**: April 15 – April 28
* **Sprint Goal**:  
  “Implement separate handling of important tasks and introduce an archive for completed tasks to keep the main list clean.”

**Selected Product Backlog Items (PBIs)**

1. **Important Tasks** (3 SP)
   * *As a user, I want to differentiate important tasks from regular tasks so that I can focus on high-priority items.*
2. **Archive / History** (5 SP)
   * *As a user, I want completed tasks to be moved to an archive so that I can keep a record without cluttering my main to-do list.*

**Total**: 8 SP

**Sprint Backlog & Tasks**

**PBI 1: Important Tasks (3 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 1.1: UI for Important Tasks | Add a separate input + button for “Important Tasks”; create a new <ul> or section on index.html. | 1 SP | Done |
| 1.2: addImportantTask() Function | Add a new array importantTaskList. Insert new tasks with a “completed” property (default false). | 1 SP | Done |
| 1.3: renderImportantTasks() | Render the “Important Tasks” list; handle toggling and deleting similarly to regular tasks. | 1 SP | Done |

**Total**: 3 SP → **All Completed**

**PBI 2: Archive / History (5 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 2.1: Archive Button + UI | Add an “Archive Completed” button for regular tasks. Create a new <div> for archived tasks. | 1 SP | Done |
| 2.2: archiveCompleted() Function | Filter out completed tasks from taskList; store them in a new historyList. | 2 SP | Done |
| 2.3: renderArchive() | Display archived tasks in read-only form; no toggles or deletes from archive. | 1 SP | Done |
| 2.4: Persist historyList | Update saveTasksToLocalStorage() and retrieveTasksFromLocalStorage() to handle archive. | 1 SP | Done |

**Total**: 5 SP → **All Completed**

**Additional Activities**

|  |  |  |
| --- | --- | --- |
| Activity | Description | Status |
| Refactoring / Integration | Ensure newly introduced “importantTaskList” and “historyList” integrate well with existing localStorage structure. | Done |
| Peer Code Review | Review changes, especially the newly introduced arrays and archive logic. | Done |
| Light Testing | Manually test adding tasks, marking them complete, archiving, retrieving from localStorage. | Done |

**Sprint 2 Review**

* **Outcome**:
  + Users can now manage **regular** vs. **important** tasks separately.
  + Users have an **Archive** button to move completed tasks out of the main list.
  + All new data (important tasks, archive list) persists in localStorage.
* **Velocity**: 8 SP completed.
* **Increment**: Enhanced ToDo List with separate **Important Tasks** section and an **Archive** feature.

# Sprint 3: UI/UX Improvements & Filtering

* **Dates**: April 29 – May 12
* **Sprint Goal**:  
  “Polish the UI for a more modern, responsive design and implement basic filtering options for easier task navigation.”

**Selected Product Backlog Items (PBIs)**

1. **UI/UX Improvements** (2 SP)
   * *As a user, I want a modern, responsive interface so that I can easily use the app on any device.*
2. **Filtering / Sorting Tasks** (5 SP)
   * *As a user, I want to filter or sort tasks by name or status so I can quickly locate tasks of interest.*

**Total**: 7 SP

**Sprint Backlog & Tasks**

**PBI 1: UI/UX Improvements (2 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 1.1: Responsive Design | Update styles.css to use media queries so layout adapts to mobile, tablet, and desktop screens. | 1 SP | Done |
| 1.2: Polish Buttons and Layout | Improve button styles, spacing, typography; ensure touch-friendly sizes for inputs and buttons. | 1 SP | Done |

**Total**: 2 SP → **All Completed**

**PBI 2: Filtering / Sorting Tasks (5 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 2.1: Add Filter Controls (UI) | Add dropdown or buttons to filter tasks: all, completed, incomplete. | 1 SP | Done |
| 2.2: Implement applyFilter() Function | When a filter is selected, only show tasks that match the criteria (e.g., completed). | 2 SP | Done |
| 2.3: Add Sort Option (Name/Date) | Possibly store creation time; allow sorting tasks alphabetically or by creation date. | 1 SP | Done |
| 2.4: Integrate Sorting & Filtering | Ensure renderTasks() (and renderImportantTasks() if relevant) can handle both filter + sort combos. | 1 SP | Done |

**Total**: 5 SP → **All Completed**

**Additional Activities**

|  |  |  |
| --- | --- | --- |
| Activity | Description | Status |
| Code Review | Verify changes in script.js to ensure filtering/sorting does not break existing logic. | Done |
| Mobile Testing | Check the site on a phone or tablet for UI responsiveness. | Done |
| Minor Refactoring | Possibly rename or reorganize code for clarity. | Done |

**Sprint 3 Review**

* **Outcome**:
  + The UI is now **more modern** and **responsive**.
  + Users can **filter** tasks by status (all, completed, incomplete) and **sort** them by name or creation date.
* **Velocity**: 7 SP completed.
* **Increment**: Users have a polished interface and improved navigation of tasks, building on Sprints 1 and 2.

# Sprint 4: Accessibility & Prep for Sync

* **Dates**: May 13 – May 26
* **Sprint Goal**:  
  “Ensure the application is accessible to all users and lay groundwork for potential sync/sharing features.”

**Selected Product Backlog Items (PBIs)**

1. **Accessibility** (3 SP)
   * *As a user with accessibility needs, I want the web app to be navigable by screen readers and keyboard so I can manage tasks without barriers.*
2. *(Optional Incomplete)* **Share / Sync Feature** (8 SP) – **Partially** begun for planning only
   * *As a user, I want to share or sync tasks across devices or with other users so that I can collaborate on tasks easily.*
   * In this sprint, the team may only start the analysis/prep work, not the full implementation.

**Total**: 3 SP fully planned + partial discovery for the sync epic.

**Sprint Backlog & Tasks**

**PBI 1: Accessibility (3 SP)**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 1.1: Semantic HTML Audit | Ensure lists use <ul>/<li> properly, buttons are real <button> elements. | 1 SP | Done |
| 1.2: ARIA & Keyboard Navigation | Add ARIA roles/labels, verify tab order, and visually highlight focus states. | 1 SP | Done |
| 1.3: Screen Reader Testing | Use a tool like NVDA or VoiceOver to confirm tasks are read out correctly. | 1 SP | Done |

**Total**: 3 SP → **All Completed**

**PBI 2: Share / Sync Feature (8 SP) – Not Fully in Scope This Sprint**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Description | Estimate | Status |
| 2.1: Discovery & Requirements | Brainstorm approaches (e.g., backend API vs. third-party sync). Gather user stories & constraints. | 2 SP | Done (Partial) |
| 2.2: Basic Architectural Plan | Decide on technology stack for future real-time or near real-time syncing. | 2 SP | Done (Partial) |
| Remaining tasks | Actual implementation (API integration, user authentication, real-time updates, etc.). | 4 SP | To Do (Next) |

Since the team decided only to **start** the discovery and planning for the sync feature, they have partially completed 4 out of 8 SP. The **remaining** 4 SP would be scheduled for a **subsequent sprint** once the architectural approach is finalized.

**Additional Activities**

|  |  |  |
| --- | --- | --- |
| Activity | Description | Status |
| Code/Design Review | Validate the accessibility changes are thoroughly integrated. | Done |
| Documentation | Update the project README with notes on accessibility improvements. | Done |
| Roadmap Discussion | Outline a roadmap for implementing full sync or share features in future. | Done |

**Sprint 4 Review**

* **Outcome**:
  + The app now **meets basic accessibility standards** (semantic tags, keyboard navigation, screen-reader checks).
  + Preliminary **research** and **technical planning** for the sync feature is done, but full functionality is **not** implemented yet.
* **Velocity**:
  + 3 SP (Accessibility) fully completed.
  + 4 SP (partial) used in analyzing the sync approach, but not delivering a shippable increment for that feature.
* **Increment**:
  + The application is fully accessible and tested with screen readers.
  + Plans are in place to tackle a real-time or multi-device sync solution in a future sprint.

# Final Notes

* **Sprint 1 → 4** show a **natural progression** of building the core product, then enhancing it with more advanced features (Important Tasks, Archive, UI/UX, Filtering, and Accessibility).
* Each sprint backlog details the **tasks** (with estimates, statuses), **PBIs**, and **team activities**.
* A new team can review these **backlogs** to understand exactly *what* was done, *when* it was done, and *why* those tasks were chosen. They can continue with future sprints focusing on **sharing/sync**, **mobile app integration**, or other backlog items not yet addressed.