

## # miniCycle - Developer Documentation

**\*\*Version\*\***: 1.275

**\*\*Target Audience\*\***: Developers, Contributors, Technical Partners

**\*\*Purpose\*\***: Comprehensive technical guide for development and integration

-----

### ## Executive Summary

miniCycle is a production-ready web application that introduces a revolutionary “cycling” approach to task management. Built with vanilla JavaScript and modern web standards, it offers automatic task reset functionality that promotes habit formation and routine establishment.

**\*\*Key Technical Highlights:\*\***

- **\*\*Zero Dependencies\*\***: Pure JavaScript implementation with no external libraries
- **\*\*Privacy-First Architecture\*\***: 100% client-side storage with no data transmission
- **\*\*Advanced Recurring System\*\***: Comprehensive scheduling rivaling enterprise solutions
- **\*\*PWA Implementation\*\***: Offline-capable with service worker and manifest
- **\*\*Dual Version Strategy\*\***: Modern ES6+ and ES5 fallback for maximum compatibility
- **\*\*Modular Design\*\***: Clean architecture enabling easy extension and customization

**\*\*Business Value:\*\***

- **\*\*Rapid Deployment\*\***: Static files only - no server infrastructure required
- **\*\*Scalable Foundation\*\***: Architecture supports feature expansion for paid versions
- **\*\*Cross-Platform\*\***: Single codebase works across all devices and browsers
- **\*\*Maintenance Efficient\*\***: Self-contained codebase with comprehensive documentation

**\*\*Technical Sophistication:\*\***

- Schema versioning with automatic data migration
- Advanced drag-and-drop with touch and mouse support
- Comprehensive theming system with unlockable rewards
- Professional error handling and recovery procedures
- Enterprise-grade notification and modal systems

This documentation provides everything needed to understand, extend, or integrate with miniCycle’s robust technical foundation.

-----

### ## Table of Contents

1. [Overview](#overview)
1. [Architecture](#architecture)
1. [Core Systems](#core-systems)
1. [API Reference](#api-reference)
1. [Data Management](#data-management)
1. [UI Components](#ui-components)
1. [Event System](#event-system)
1. [Development Guide](#development-guide)
1. [Troubleshooting](#troubleshooting)
1. [Appendix: Advanced Customization](#appendix-advanced-customization)

----

## ## Overview

miniCycle is a sophisticated web-based task management application that revolutionizes productivity through automatic task cycling. Unlike traditional task managers, miniCycle resets completed task lists to promote habit formation and routine establishment.

### ### Core Philosophy

- **\*\*Privacy-First\*\***: All data stored locally, no external servers
- **\*\*Offline-Capable\*\***: Full functionality without internet connection
- **\*\*Habit-Focused\*\***: Cycling methodology encourages consistent routines
- **\*\*Cross-Platform\*\***: Responsive design works on all devices

### ### Key Differentiators

- Automatic task reset system
- Advanced recurring task scheduling
- Multiple cycle management
- Unlockable theme system with gamification
- Comprehensive PWA implementation

----

## ## Architecture

### ### Technology Stack

...

Frontend Layer:

└─ HTML5 (Semantic structure)

- └─ CSS3 (Custom properties, Grid, Flexbox)
- └─ Vanilla JavaScript (ES6+ with ES5 fallback)
- └─ Progressive Web App (Service Worker, Manifest)

#### Data Layer:

- └─ localStorage (Primary storage)
- └─ JSON schema (Version 2.5)
- └─ Migration system (Backwards compatibility)
- └─ Export/Import (.mcy files)

#### Compatibility Layer:

- └─ Modern browsers (Chrome, Firefox, Safari, Edge)
- └─ ES5 fallback (miniCycle-lite.html)
- └─ Touch and mouse events
- └─ Responsive breakpoints

...

### ### Project Structure

...

#### miniCycle/

- └─ Core Application
  - └─ miniCycle.html # Main application entry
  - └─ miniCycle-scripts.js # Core logic (ES6+)
  - └─ miniCycle-styles.css # Main stylesheet
  - └─ manifest.json # PWA configuration
- └─ Compatibility Version
  - └─ miniCycle-lite.html # ES5 compatible entry
  - └─ miniCycle-lite-scripts.js # ES5 compatible logic
  - └─ miniCycle-lite-styles.css # Optimized styles
  - └─ manifest-lite.json # Lite PWA config
- └─ Documentation
  - └─ user-manual.html # End user guide
  - └─ user-manual-styles.css # Manual styling
  - └─ privacy.html # Privacy policy
  - └─ terms.html # Terms of service
- └─ Assets
  - └─ icons/ # PWA icons (various sizes)
  - └─ images/ # App screenshots, logos

...

-----

## ## Core Systems

### ### 1. Task Management System

#### #### Task Creation and Validation

```
``javascript
function addTask(taskText, completed = false, shouldSave = true,
    dueDate = null, highPriority = null, isLoading = false,
    remindersEnabled = false, recurring = false,
    taskId = null, recurringSettings = {}) {

    // Input validation and sanitization
    const sanitizedText = sanitizeInput(taskText);
    if (!sanitizedText) {
        showNotification("Task text cannot be empty", "error");
        return false;
    }

    // Generate unique task ID
    const id = taskId || generateUniqueId();

    // Create task object with schema version
    const task = {
        id: id,
        text: sanitizedText,
        completed: completed,
        priority: highPriority || false,
        dueDate: dueDate,
        remindersEnabled: remindersEnabled,
        recurring: recurring,
        recurringSettings: recurringSettings,
        schemaVersion: SCHEMA_VERSION,
        createdAt: new Date().toISOString(),
        completedAt: null
    };

    // Add to task list and update UI
    taskList.push(task);
    if (shouldSave) autoSave();

    return task;
}
```

```
}  
...
```

#### #### Task State Management

```
````javascript  
function handleTaskCompletionChange(checkbox) {  
    const taskElement = checkbox.closest('.task');  
    const taskId = taskElement.dataset.taskId;  
    const task = findTaskById(taskId);  
  
    if (!task) return;  
  
    // Save state for undo functionality  
    saveUndoState('task_completion');  
  
    // Update task state  
    task.completed = checkbox.checked;  
    task.completedAt = checkbox.checked ? new Date().toISOString() : null;  
  
    // Visual feedback  
    taskElement.classList.toggle('completed', checkbox.checked);  
  
    // Check for cycle completion  
    if (allTasksCompleted() && getCurrentCycle().autoReset) {  
        triggerCycleCompletion();  
    }  
  
    autoSave();  
    updateProgress();  
}  
...
```

#### ### 2. Cycle Management System

##### #### Cycle Types and Behavior

```
````javascript  
const CycleTypes = {  
    AUTO_RESET: {  
        behavior: 'auto_reset',  
        resetDelay: 3000, // 3 seconds  
        deleteCompleted: false,  
        showCompleteButton: false
```

```

},
MANUAL_RESET: {
  behavior: 'manual_reset',
  resetDelay: null,
  deleteCompleted: false,
  showCompleteButton: true
},
TODO_LIST: {
  behavior: 'delete_completed',
  resetDelay: null,
  deleteCompleted: true,
  showCompleteButton: false
}
};
...

```

#### ##### Cycle Operations

```

```javascript
function createNewMiniCycle(name, autoReset = true, deleteChecked = false) {
  // Validate cycle name
  if (miniCycleStorage[name]) {
    showNotification("A cycle with that name already exists", "error");
    return false;
  }

  // Create new cycle structure
  const newCycle = {
    title: name,
    tasks: [],
    recurringTemplates: {},
    autoReset: autoReset,
    deleteCheckedTasks: deleteChecked,
    cycleCount: 0,
    createdAt: new Date().toISOString(),
    lastModified: new Date().toISOString()
  };

  // Save to storage
  miniCycleStorage[name] = newCycle;
  localStorage.setItem('miniCycleStorage', JSON.stringify(miniCycleStorage));

  showNotification(`Created new cycle: ${name}`, "success");
  return true;
}

```

```

}

function switchToMiniCycle(cycleName) {
  if (!miniCycleStorage[cycleName]) {
    showNotification("Cycle not found", "error");
    return false;
  }

  // Save current state
  autoSave();

  // Switch active cycle
  localStorage.setItem('activeMiniCycle', cycleName);

  // Load new cycle
  loadMiniCycle();

  showNotification(`Switched to: ${cycleName}`, "info");
  return true;
}
...

```

### ### 3. Recurring Tasks System

#### #### Recurring Settings Schema

```

````javascript
const RecurringSettings = {
  frequency: "hourly|daily|weekly|monthly|yearly",
  indefinitely: true, // or false with count
  count: null,       // number of occurrences if not indefinite

  // Time settings
  time: {
    hour: 9,         // 1-12 or 0-23
    minute: 0,       // 0-59
    meridiem: "AM",  // "AM" or "PM"
    military: false  // 24-hour format
  },

  // Weekly settings
  weekly: {
    days: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
  },
}

```

```

// Monthly settings
monthly: {
  days: [1, 15, 30] // Days of month
},

// Yearly settings
yearly: {
  months: [1, 6, 12], // January, June, December
  daysByMonth: {
    1: [1], // January 1st
    6: [15], // June 15th
    12: [25] // December 25th
  }
},

// Specific dates override
specificDates: {
  enabled: false,
  dates: ["2025-12-25", "2025-07-04"]
}
};
...

```

#### #### Recurring Task Logic

```

```javascript
function shouldTaskRecurNow(settings, now = new Date()) {
  const { frequency, time, indefinitely, count } = settings;

  // Check if task has exceeded occurrence limit
  if (!indefinitely && settings.occurrenceCount >= count) {
    return false;
  }

  // Parse target time
  const targetTime = parseTimeSettings(time);
  const currentTime = {
    hour: now.getHours(),
    minute: now.getMinutes()
  };

  // Check if current time matches target time
  if (!timeMatches(currentTime, targetTime)) {

```



```

    return false;
}

// Frequency-specific checks
switch (frequency) {
    case 'hourly':
        return currentTime.minute === targetTime.minute;

    case 'daily':
        return true; // Time already checked above

    case 'weekly':
        const dayName = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'][now.getDay()];
        return settings.weekly.days.includes(dayName);

    case 'monthly':
        const dayOfMonth = now.getDate();
        return settings.monthly.days.includes(dayOfMonth);

    case 'yearly':
        const month = now.getMonth() + 1; // 0-based to 1-based
        const day = now.getDate();
        return settings.yearly.months.includes(month) &&
            settings.yearly.daysByMonth[month]?.includes(day);

    default:
        return false;
}
}

// Background monitoring system
function watchRecurringTasks() {
    const now = new Date();
    const activeCycle = getCurrentCycle();

    Object.entries(activeCycle.recurringTemplates).forEach(([taskId, template]) => {
        if (shouldTaskRecurNow(template.recurringSettings, now)) {
            // Check if task already exists (prevent duplicates)
            const existingTask = findTaskById(taskId);
            if (!existingTask || existingTask.completed) {
                recreateRecurringTask(template);
            }
        }
    });
}

```

```
}
```

```
// Run every 30 seconds  
setInterval(watchRecurringTasks, 30000);  
````
```

```
-----
```

## ## API Reference

### ### Core Task Functions

#### ##### `addTask(options)`

Creates a new task with comprehensive options.

```
````javascript  
// Basic usage  
addTask("Complete project documentation");  
  
// Advanced usage with all options  
addTask("Daily standup meeting", false, true, "2025-09-16T09:00:00Z", true, false, true, true,  
null, {  
  frequency: "daily",  
  indefinitely: true,  
  time: { hour: 9, minute: 0, meridiem: "AM" }  
});  
````
```

#### ##### `editTask(taskId, newText, options)`

Modifies existing task properties.

```
````javascript  
editTask("task-123", "Updated task description", {  
  priority: true,  
  dueDate: "2025-09-20T17:00:00Z"  
});  
````
```

#### ##### `deleteTask(taskId)`

Removes task and cleans up related data.

```
```javascript
deleteTask("task-123");
// Automatically saves state for undo and updates UI
```
```

### ### Cycle Management Functions

```
##### `createNewMiniCycle(name, type)`
```

Creates a new task cycle with specified behavior.

```
```javascript
// Auto-reset cycle
createNewMiniCycle("Morning Routine", true, false);

// Manual reset cycle
createNewMiniCycle("Project Tasks", false, false);

// To-do list mode
createNewMiniCycle("Shopping List", false, true);
```
```

```
##### `exportMiniCycle(cycleName)`
```

Exports cycle as downloadable .mcyf file.

```
```javascript
exportMiniCycle("Morning Routine");
// Downloads: morning-routine.mcyf
```
```

### ### Storage Functions

```
##### `autoSave(overrideTaskList)`
```

Saves current application state to localStorage.

```
```javascript
// Save current state
autoSave();

// Save specific task list
autoSave(customTaskList);
```
```

```
#### `loadMiniCycle(cycleName)`
```

Loads specified cycle or active cycle from storage.

```
```javascript
loadMiniCycle(); // Load active cycle
loadMiniCycle("Work Tasks"); // Load specific cycle
```
```

-----

```
## Data Management
```

```
### Schema Evolution System
```

```
#### Current Schema (Version 2.5)
```

```
```javascript
const SCHEMA_VERSION = 2.5;

// Task schema
const TaskSchema = {
  id: "string",      // Unique identifier
  text: "string",    // Task description (max 50 chars)
  completed: "boolean", // Completion state
  priority: "boolean", // High priority flag
  dueDate: "string|null", // ISO 8601 date string
  remindersEnabled: "boolean",
  recurring: "boolean",
  recurringSettings: "object",
  schemaVersion: "number",
  createdAt: "string", // ISO 8601 timestamp
  completedAt: "string|null"
};
```

```
// Cycle schema
const CycleSchema = {
  title: "string",
  tasks: "array",
  recurringTemplates: "object",
  autoReset: "boolean",
  deleteCheckedTasks: "boolean",
  cycleCount: "number",
```

```
    createdAt: "string",
    lastModified: "string"
  };
  ...
```

#### #### Migration System

```
````javascript
function migrateTask(task) {
  let migrated = { ...task };

  // Migrate from schema 1.0 to 2.0
  if (!task.schemaVersion || task.schemaVersion < 2.0) {
    migrated.priority = false;
    migrated.dueDate = null;
    migrated.remindersEnabled = false;
    migrated.schemaVersion = 2.0;
  }

  // Migrate from schema 2.0 to 2.5
  if (migrated.schemaVersion < 2.5) {
    migrated.recurring = false;
    migrated.recurringSettings = {};
    migrated.createdAt = migrated.createdAt || new Date().toISOString();
    migrated.completedAt = null;
    migrated.schemaVersion = 2.5;
  }

  return migrated;
}
...

```

#### ### Storage Structure

```
````javascript
// localStorage keys and structure
const StorageKeys = {
  MINI_CYCLE_STORAGE: 'miniCycleStorage',
  ACTIVE_MINI_CYCLE: 'activeMiniCycle',
  REMINDERS_SETTINGS: 'remindersSettings',
  THEME_SETTINGS: 'selectedTheme',
  APP_SETTINGS: 'appSettings'
};

```

```

// Example storage content
const ExampleStorage = {
  miniCycleStorage: {
    "Morning Routine": {
      title: "Morning Routine",
      tasks: [
        {
          id: "task-001",
          text: "Drink water",
          completed: false,
          priority: false,
          dueDate: null,
          remindersEnabled: false,
          recurring: true,
          recurringSettings: {
            frequency: "daily",
            indefinitely: true,
            time: { hour: 7, minute: 0, meridiem: "AM" }
          },
          schemaVersion: 2.5,
          createdAt: "2025-09-16T06:00:00Z",
          completedAt: null
        }
      ],
      recurringTemplates: {},
      autoReset: true,
      deleteCheckedTasks: false,
      cycleCount: 12,
      createdAt: "2025-09-01T06:00:00Z",
      lastModified: "2025-09-16T06:00:00Z"
    }
  },
  activeMiniCycle: "Morning Routine",
  remindersSettings: {
    enabled: true,
    indefinite: false,
    dueDatesReminders: true,
    repeatCount: 3,
    frequencyValue: 30,
    frequencyUnit: "minutes"
  }
};
...

```

-----

## ## UI Components

### #### Modal System

#### ##### Settings Modal

```
```javascript
function setupSettingsModal() {
  const settingsModal = document.getElementById('settingsModal');

  // Dark mode toggle
  const darkModeToggle = document.getElementById('darkModeToggle');
  darkModeToggle.addEventListener('change', (e) => {
    toggleDarkMode(e.target.checked);
  });

  // Move arrows toggle
  const moveArrowsToggle = document.getElementById('moveArrowsToggle');
  moveArrowsToggle.addEventListener('change', (e) => {
    toggleMoveArrows(e.target.checked);
  });

  // Three-dot menu toggle
  const threeDotToggle = document.getElementById('threeDotMenuToggle');
  threeDotToggle.addEventListener('change', (e) => {
    toggleThreeDotMenu(e.target.checked);
  });
}
```
```

#### ##### Recurring Tasks Panel

```
```javascript
function setupRecurringPanel() {
  const panel = document.getElementById('recurringPanel');

  // Frequency selection
  const frequencySelect = document.getElementById('frequencySelect');
  frequencySelect.addEventListener('change', (e) => {
    updateRecurringOptions(e.target.value);
  });
}
```

```

// Time picker
const timePicker = {
  hour: document.getElementById('hourSelect'),
  minute: document.getElementById('minuteSelect'),
  meridiem: document.getElementById('meridiemSelect')
};

Object.values(timePicker).forEach(select => {
  select.addEventListener('change', updateTimePreview);
});

// Day selection for weekly recurring
const dayCheckboxes = document.querySelectorAll('.day-checkbox');
dayCheckboxes.forEach(checkbox => {
  checkbox.addEventListener('change', updateSelectedDays);
});
}
...

```

### ### Notification System

```

````javascript
function showNotification(message, type = 'default', duration = 3000) {
  const container = document.getElementById('notification-container');

  const notification = document.createElement('div');
  notification.className = `notification notification-${type}`;
  notification.innerHTML = `
    <span class="notification-message">${message}</span>
    <button class="notification-close">&times;</button>
  `;

  // Auto-dismiss timer
  const timer = setTimeout(() => {
    removeNotification(notification);
  }, duration);

  // Manual close button
  const closeBtn = notification.querySelector('.notification-close');
  closeBtn.addEventListener('click', () => {
    clearTimeout(timer);
    removeNotification(notification);
  });
}

```



```

container.appendChild(notification);

// Animate in
requestAnimationFrame(() => {
  notification.classList.add('notification-visible');
});
}

```

```

// Notification types with styling
const NotificationTypes = {
  default: 'blue background, white text',
  success: 'green background, white text',
  error: 'red background, white text',
  warning: 'orange background, white text',
  info: 'gray background, white text',
  recurring: 'purple background, white text'
};
...

```

### ### Drag and Drop System

```

````javascript
function initializeDragAndDrop(taskElement) {
  let draggedElement = null;
  let initialY = 0;
  let currentY = 0;

  // Touch events for mobile
  taskElement.addEventListener('touchstart', (e) => {
    draggedElement = taskElement;
    initialY = e.touches[0].clientY;
    taskElement.classList.add('dragging');
  });

  taskElement.addEventListener('touchmove', (e) => {
    if (!draggedElement) return;

    e.preventDefault();
    currentY = e.touches[0].clientY;
    const diffY = currentY - initialY;

    // Visual feedback
    taskElement.style.transform = `translateY(${diffY}px)`;
  });
}

```

```

// Find drop target
const afterElement = getDragAfterElement(container, currentY);
if (afterElement) {
  container.insertBefore(draggedElement, afterElement);
} else {
  container.appendChild(draggedElement);
}
});

taskElement.addEventListener('touchend', () => {
  if (draggedElement) {
    draggedElement.style.transform = "";
    draggedElement.classList.remove('dragging');
    draggedElement = null;

    // Save new order
    updateTaskOrder();
    autoSave();
  }
});

// Mouse events for desktop
taskElement.addEventListener('dragstart', (e) => {
  draggedElement = taskElement;
  e.dataTransfer.effectAllowed = 'move';
});

taskElement.addEventListener('dragover', (e) => {
  e.preventDefault();
  e.dataTransfer.dropEffect = 'move';
});

taskElement.addEventListener('drop', (e) => {
  e.preventDefault();
  if (draggedElement && draggedElement !== taskElement) {
    const container = taskElement.parentNode;
    const afterElement = getDragAfterElement(container, e.clientY);

    if (afterElement) {
      container.insertBefore(draggedElement, afterElement);
    } else {
      container.appendChild(draggedElement);
    }
  }
});

```

```

        updateTaskOrder();
        autoSave();
    }
});
}
...

```

-----

## ## Event System

### ### Global Event Handlers

```

````javascript
// Initialize global event listeners
function initializeGlobalEvents() {
    // Keyboard shortcuts
    document.addEventListener('keydown', (e) => {
        // Undo functionality
        if ((e.ctrlKey || e.metaKey) && e.key === 'z' && !e.shiftKey) {
            e.preventDefault();
            performUndo();
        }

        // Redo functionality
        if ((e.ctrlKey || e.metaKey) && (e.key === 'y' || (e.key === 'z' && e.shiftKey))) {
            e.preventDefault();
            performRedo();
        }

        // Close modals with Escape
        if (e.key === 'Escape') {
            closeAllModals();
        }

        // Quick add task with Ctrl+Enter
        if ((e.ctrlKey || e.metaKey) && e.key === 'Enter') {
            const taskInput = document.getElementById('taskInput');
            if (taskInput.value.trim()) {
                e.preventDefault();
                addTaskFromInput();
            }
        }
    });
}

```

```

// Window events
window.addEventListener('beforeunload', (e) => {
  // Auto-save before page unload
  autoSave();
});

window.addEventListener('resize', () => {
  // Responsive adjustments
  adjustLayoutForScreenSize();
});

// Visibility change (tab switching)
document.addEventListener('visibilitychange', () => {
  if (!document.hidden) {
    // Check for recurring tasks when tab becomes visible
    watchRecurringTasks();
  }
});
...

```

### ### Custom Event System

```

````javascript
// Event dispatcher for internal communication
const EventSystem = {
  listeners: {},

  on(event, callback) {
    if (!this.listeners[event]) {
      this.listeners[event] = [];
    }
    this.listeners[event].push(callback);
  },

  emit(event, data) {
    if (this.listeners[event]) {
      this.listeners[event].forEach(callback => callback(data));
    }
  },

  off(event, callback) {
    if (this.listeners[event]) {

```

```

        this.listeners[event] = this.listeners[event].filter(cb => cb !== callback);
    }
}
};

```

// Usage examples

```

EventSystem.on('taskCompleted', (task) => {
    updateProgress();
    checkCycleCompletion();
    showCompletionAnimation(task);
});

```

```

EventSystem.on('cycleCompleted', (cycle) => {
    incrementCycleCount();
    checkMilestoneUnlocks();
    triggerCelebration();
});

```

// Emit events

```

function completeTask(taskId) {
    const task = findTaskById(taskId);
    task.completed = true;
    EventSystem.emit('taskCompleted', task);
}
...

```

-----

## ## Development Guide

### #### Setup and Installation

```

```bash
# No build process required - serve static files
# For local development:
python -m http.server 8000
# OR
npx serve .
# OR
php -S localhost:8000
```

```

### #### Development Workflow

```

````javascript
// 1. Enable debug mode
const DEBUG_MODE = true;

// 2. Use console logging for development
function debugLog(message, data = null) {
  if (DEBUG_MODE) {
    console.log(`[miniCycle Debug] ${message}`, data);
  }
}

// 3. Testing utilities
const DevTools = {
  // Simulate recurring task trigger
  triggerRecurringTask(taskId) {
    const template = getCurrentCycle().recurringTemplates[taskId];
    if (template) {
      recreateRecurringTask(template);
    }
  },

  // Force milestone unlock
  unlockMilestone(milestone) {
    const currentCount = getTotalCycleCount();
    setTotalCycleCount(milestone.requirement);
    checkMilestoneUnlocks();
    setTotalCycleCount(currentCount);
  },

  // Reset all data
  factoryReset() {
    if (confirm('This will delete ALL data. Continue?')) {
      localStorage.clear();
      location.reload();
    }
  }
};

// Expose dev tools in development
if (DEBUG_MODE) {
  window.DevTools = DevTools;
}
...

```

### ### Testing Strategies

```
```javascript
// Unit testing approach (manual)
function testTaskCreation() {
  const originalTaskCount = taskList.length;

  addTask("Test task");

  console.assert(taskList.length === originalTaskCount + 1, "Task not added");
  console.assert(taskList[taskList.length - 1].text === "Test task", "Task text incorrect");

  console.log("✅ Task creation test passed");
}

// Integration testing
function testCycleCompletion() {
  // Create test cycle with auto-reset
  createNewMiniCycle("Test Cycle", true, false);
  switchToMiniCycle("Test Cycle");

  // Add test tasks
  addTask("Task 1");
  addTask("Task 2");

  // Complete all tasks
  taskList.forEach(task => task.completed = true);

  // Trigger cycle completion check
  setTimeout(() => {
    console.assert(taskList.every(task => !task.completed), "Tasks not reset");
    console.log("✅ Cycle completion test passed");
  }, 4000); // Wait for auto-reset delay
}
...
```
```

### ### Browser Compatibility Testing

```
```javascript
// Feature detection
function checkBrowserCompatibility() {
  const features = {
    localStorage: typeof(Storage) !== "undefined",
    flexbox: CSS.supports('display', 'flex'),
  }
}
```

```

    grid: CSS.supports('display', 'grid'),
    customProperties: CSS.supports('color', 'var(--color)'),
    serviceWorker: 'serviceWorker' in navigator,
    touchEvents: 'ontouchstart' in window
  };

  console.table(features);

  const incompatible = Object.entries(features)
    .filter(([feature, supported]) => !supported)
    .map(([feature]) => feature);

  if (incompatible.length > 0) {
    console.warn('Unsupported features:', incompatible);
    showNotification('Some features may not work in this browser', 'warning');
  }
}
...

```

-----

## ## Troubleshooting

### ### Common Issues and Solutions

#### #### Storage Issues

```

```javascript
// Problem: localStorage quota exceeded
function handleStorageQuotaExceeded() {
  try {
    // Attempt to save
    autoSave();
  } catch (error) {
    if (error.name === 'QuotaExceededError') {
      // Clean up old data
      cleanupOldCycles();

      // Retry save with essential data only
      const essentialData = getEssentialData();
      saveEssentialData(essentialData);

      showNotification('Storage limit reached. Old cycles cleaned up.', 'warning');
    }
  }
}

```



```
}  
}
```

// Problem: Corrupted data in localStorage

```
function validateAndRepairStorage() {  
  try {  
    const data = JSON.parse(localStorage.getItem('miniCycleStorage') || '{}');  
  
    // Validate structure  
    Object.entries(data).forEach(([cycleName, cycle]) => {  
      if (!cycle.tasks || !Array.isArray(cycle.tasks)) {  
        console.warn(`Repairing cycle: ${cycleName}`);  
        cycle.tasks = [];  
      }  
  
      // Validate tasks  
      cycle.tasks = cycle.tasks.filter(task => {  
        return task && typeof task.id === 'string' && typeof task.text === 'string';  
      });  
  
      // Migrate old schema  
      cycle.tasks = cycle.tasks.map(migrateTask);  
    });  
  
    // Save repaired data  
    localStorage.setItem('miniCycleStorage', JSON.stringify(data));  
    return true;  
  
  } catch (error) {  
    console.error('Storage repair failed:', error);  
    return false;  
  }  
}  
...
```

#### UI Issues

```javascript

// Problem: Drag and drop not working on mobile

```
function fixMobileInteractions() {  
  // Ensure touch events are properly handled  
  const taskElements = document.querySelectorAll('.task');  
  
  taskElements.forEach(element => {
```

```

// Remove existing listeners
element.removeEventListener('touchstart', handleTouchStart);
element.removeEventListener('touchmove', handleTouchMove);
element.removeEventListener('touchend', handleTouchEnd);

// Re-add with proper options
element.addEventListener('touchstart', handleTouchStart, { passive: false });
element.addEventListener('touchmove', handleTouchMove, { passive: false });
element.addEventListener('touchend', handleTouchEnd, { passive: false });
});
}

```

```

// Problem: Modal not closing properly
function ensureModalCleanup() {
  // Force close all modals
  const modals = document.querySelectorAll('.modal');
  modals.forEach(modal => {
    modal.style.display = 'none';
    modal.classList.remove('active');
  });

  // Reset body scroll
  document.body.style.overflow = "";

  // Clear any modal backdrop
  const backdrops = document.querySelectorAll('.modal-backdrop');
  backdrops.forEach(backdrop => backdrop.remove());
}
...

```

#### #### Performance Issues

```

```javascript
// Problem: Slow rendering with many tasks
function optimizeTaskRendering() {
  const container = document.getElementById('taskList');

  // Use document fragment for batch updates
  const fragment = document.createDocumentFragment();

  taskList.forEach(task => {
    const taskElement = createTaskElement(task);
    fragment.appendChild(taskElement);
  });
}

```

```

// Single DOM update
container.innerHTML = "";
container.appendChild(fragment);
}

// Problem: Memory leaks from event listeners
function cleanupEventListeners() {
  // Remove all task-specific listeners
  const taskElements = document.querySelectorAll('.task');
  taskElements.forEach(element => {
    // Clone element to remove all listeners
    const newElement = element.cloneNode(true);
    element.parentNode.replaceChild(newElement, element);

    // Re-add essential listeners
    initializeTaskElement(newElement);
  });
}
...

```

### ### Debug Tools

```

````javascript
// Diagnostic functions
const Diagnostics = {
  // Check data integrity
  validateData() {
    const storage = JSON.parse(localStorage.getItem('miniCycleStorage') || '{}');
    const issues = [];

    Object.entries(storage).forEach(([cycleName, cycle]) => {
      if (!cycle.tasks) {
        issues.push(`${cycleName}: Missing tasks array`);
      }

      if (cycle.tasks) {
        cycle.tasks.forEach((task, index) => {
          if (!task.id) {
            issues.push(`${cycleName}[${index}]: Missing task ID`);
          }
          if (!task.text) {
            issues.push(`${cycleName}[${index}]: Missing task text`);
          }
        });
      }
    });
  }
};

```

```

    });
  }
});

console.log('Data validation results:', issues.length === 0 ? 'All good!' : issues);
return issues;
},

// Performance monitoring
measurePerformance(functionName, fn) {
  const start = performance.now();
  const result = fn();
  const end = performance.now();

  console.log(`${functionName} took ${(end - start).toFixed(2)}ms`);
  return result;
},

// Export diagnostic report
exportDiagnostics() {
  const report = {
    timestamp: new Date().toISOString(),
    userAgent: navigator.userAgent,
    screenSize: `${screen.width}x${screen.height}`,
    windowSize: `${window.innerWidth}x${window.innerHeight}`,
    localStorage: {
      available: typeof(Storage) !== "undefined",
      usage: JSON.stringify(localStorage).length
    },
    dataValidation: this.validateData(),
    cycleCount: Object.keys(JSON.parse(localStorage.getItem('miniCycleStorage') ||
'{}')).length,
    activeTheme: document.body.className
  };

  // Download as JSON
  const blob = new Blob([JSON.stringify(report, null, 2)], { type: 'application/json' });
  const url = URL.createObjectURL(blob);
  const a = document.createElement('a');
  a.href = url;
  a.download = `minicycle-diagnostics-${Date.now()}.json`;
  a.click();
  URL.revokeObjectURL(url);
}

```

```
};

// Expose diagnostics in development
if (typeof window !== 'undefined') {
  window.MiniCycleDiagnostics = Diagnostics;
}
...
```

-----

## ## Appendix: Advanced Customization

\*This section covers advanced topics for extending miniCycle functionality. These features are optional and intended for specialized use cases.\*

### ### A. Custom Storage Backends

#### #### Interface Definition

```
```javascript
// Interface for custom storage
class StorageBackend {
  async save(key, data) {
    throw new Error('save() must be implemented');
  }

  async load(key) {
    throw new Error('load() must be implemented');
  }

  async delete(key) {
    throw new Error('delete() must be implemented');
  }
}
...```
```

#### #### IndexedDB Implementation

```
```javascript
class IndexedDBBackend extends StorageBackend {
  constructor(dbName = 'miniCycleDB', version = 1) {
    super();
    this.dbName = dbName;
    this.version = version;
  }
}
...```
```

```

    this.db = null;
  }

  async init() {
    return new Promise((resolve, reject) => {
      const request = indexedDB.open(this.dbName, this.version);

      request.onerror = () => reject(request.error);
      request.onsuccess = () => {
        this.db = request.result;
        resolve(this.db);
      };

      request.onupgradeneeded = (e) => {
        const db = e.target.result;
        if (!db.objectStoreNames.contains('miniCycle')) {
          db.createObjectStore('miniCycle', { keyPath: 'key' });
        }
      };
    });
  }

  async save(key, data) {
    if (!this.db) await this.init();

    const transaction = this.db.transaction(['miniCycle'], 'readwrite');
    const store = transaction.objectStore('miniCycle');

    return new Promise((resolve, reject) => {
      const request = store.put({ key, data });
      request.onsuccess = () => resolve();
      request.onerror = () => reject(request.error);
    });
  }

  async load(key) {
    if (!this.db) await this.init();

    const transaction = this.db.transaction(['miniCycle'], 'readonly');
    const store = transaction.objectStore('miniCycle');

    return new Promise((resolve, reject) => {
      const request = store.get(key);
      request.onsuccess = () => resolve(request.result?.data || null);
    });
  }

```

```

        request.onerror = () => reject(request.error);
    });
}

async delete(key) {
    if (!this.db) await this.init();

    const transaction = this.db.transaction(['miniCycle'], 'readwrite');
    const store = transaction.objectStore('miniCycle');

    return new Promise((resolve, reject) => {
        const request = store.delete(key);
        request.onsuccess = () => resolve();
        request.onerror = () => reject(request.error);
    });
}
}

```

```

// Usage
const customStorage = new IndexedDBBackend();

// Replace localStorage functions
async function autoSaveAsync(overrideTaskList) {
    const data = overrideTaskList || getCurrentCycleData();
    await customStorage.save('currentCycle', data);
}
...

```

## #### B. Plugin System Architecture

### ##### Plugin Base Class

```

```javascript
class MiniCyclePlugin {
    constructor(name, version) {
        this.name = name;
        this.version = version;
        this.enabled = false;
    }

    // Lifecycle methods
    onLoad() {}
    onUnload() {}
    onTaskAdded(task) {}
}

```

```

onTaskCompleted(task) {}
onCycleCompleted(cycle) {}

// UI extension points
addMenuItem(label, callback) {
  PluginManager.addMenuItem(this.name, label, callback);
}

addTaskAction(icon, label, callback) {
  PluginManager.addTaskAction(this.name, icon, label, callback);
}
}
...

#### Plugin Manager

```javascript
const PluginManager = {
  plugins: new Map(),

  register(plugin) {
    this.plugins.set(plugin.name, plugin);
    console.log(`Plugin registered: ${plugin.name} v${plugin.version}`);
  },

  enable(pluginName) {
    const plugin = this.plugins.get(pluginName);
    if (plugin) {
      plugin.enabled = true;
      plugin.onLoad();
      this.updateUI();
    }
  },

  disable(pluginName) {
    const plugin = this.plugins.get(pluginName);
    if (plugin) {
      plugin.enabled = false;
      plugin.onUnload();
      this.updateUI();
    }
  },

  triggerEvent(eventName, data) {

```



```

    this.plugins.forEach(plugin => {
      if (plugin.enabled && typeof plugin[eventName] === 'function') {
        plugin[eventName](data);
      }
    });
  }
};
...

```

#### #### Example Plugin: Time Tracking

```

````javascript
class TimeTrackingPlugin extends MiniCyclePlugin {
  constructor() {
    super('TimeTracking', '1.0.0');
    this.startTimes = new Map();
  }

  onLoad() {
    // Add timer button to tasks
    this.addAction('🕒', 'Start Timer', (taskId) => {
      this.startTimer(taskId);
    });
  }

  onTaskCompleted(task) {
    if (this.startTimes.has(task.id)) {
      const startTime = this.startTimes.get(task.id);
      const duration = Date.now() - startTime;

      // Store time data
      this.saveTimeData(task.id, duration);
      this.startTimes.delete(task.id);
    }
  }

  startTimer(taskId) {
    this.startTimes.set(taskId, Date.now());
    showNotification('Timer started for task', 'info');
  }

  saveTimeData(taskId, duration) {
    const timeData = JSON.parse(localStorage.getItem('timeTrackingData') || '{}');
    if (!timeData[taskId]) timeData[taskId] = [];
  }
}

```

```

        timeData[taskId].push({
            duration: duration,
            date: new Date().toISOString()
        });

        localStorage.setItem('timeTrackingData', JSON.stringify(timeData));
    }
}

```

```

// Register and enable plugin
const timeTracker = new TimeTrackingPlugin();
PluginManager.register(timeTracker);
PluginManager.enable('TimeTracking');
...

```

### ### C. Custom Theme Creation

#### #### Theme Definition

```

```javascript
// Define theme CSS
const customTheme = {
    name: 'Ocean Breeze',
    unlockRequirement: 25, // cycles
    cssClass: 'theme-ocean-breeze'
};

// Add CSS rules
const themeStyles = `
body.theme-ocean-breeze {
    background: linear-gradient(135deg, #006994, #00a8cc);
    color: #ffffff;
}

body.theme-ocean-breeze .task {
    background: rgba(255, 255, 255, 0.1);
    border: 1px solid rgba(255, 255, 255, 0.2);
}

body.theme-ocean-breeze .task.completed {
    background: rgba(0, 255, 128, 0.2);
}
`;

```

```

// Register theme
function registerCustomTheme(theme) {
  // Add to theme list
  themeList.push(theme);

  // Add CSS to document
  const styleSheet = document.createElement('style');
  styleSheet.textContent = themeStyles;
  document.head.appendChild(styleSheet);

  // Update theme selector UI
  updateThemeSelector();
}
...

```

#### #### D. Data Recovery Procedures

##### ##### Emergency Backup System

```

````javascript
const DataRecovery = {
  // Create emergency backup
  createEmergencyBackup() {
    const allData = {
      storage: localStorage.getItem('miniCycleStorage'),
      active: localStorage.getItem('activeMiniCycle'),
      settings: localStorage.getItem('appSettings'),
      reminders: localStorage.getItem('remindersSettings'),
      theme: localStorage.getItem('selectedTheme')
    };

    const backup = JSON.stringify(allData);
    const blob = new Blob([backup], { type: 'application/json' });
    const url = URL.createObjectURL(blob);

    const a = document.createElement('a');
    a.href = url;
    a.download = `minicycle-emergency-backup-${Date.now()}.json`;
    a.click();

    URL.revokeObjectURL(url);
    showNotification('Emergency backup created', 'success');
  },
}

```

```

// Restore from emergency backup
restoreFromBackup(file) {
  const reader = new FileReader();
  reader.onload = (e) => {
    try {
      const backup = JSON.parse(e.target.result);

      // Restore each data type
      Object.entries(backup).forEach(([key, value]) => {
        if (value) {
          const storageKey = this.getStorageKeyMapping(key);
          localStorage.setItem(storageKey, value);
        }
      });

      showNotification('Backup restored successfully', 'success');
      setTimeout(() => location.reload(), 2000);

    } catch (error) {
      showNotification('Backup file is corrupted', 'error');
      console.error('Restore failed:', error);
    }
  };

  reader.readAsText(file);
},

getStorageKeyMapping(backupKey) {
  const mapping = {
    storage: 'miniCycleStorage',
    active: 'activeMiniCycle',
    settings: 'appSettings',
    reminders: 'remindersSettings',
    theme: 'selectedTheme'
  };
  return mapping[backupKey] || backupKey;
}
};
...

```

-----

## Conclusion

This developer documentation provides a comprehensive guide to understanding, extending, and maintaining the miniCycle application. The modular architecture and extensive customization options make it possible to adapt the application for various use cases while maintaining the core cycling methodology that makes miniCycle unique.

For additional technical details, refer to the comprehensive technical documentation. For user-facing information, consult the user manual.

#### ### Quick Reference Links

- **User Manual**: `user-manual.html`
- **Comprehensive Documentation**: Technical specification document
- **Privacy Policy**: `privacy.html`
- **Terms of Service**: `terms.html`

#### ### Support and Contributing

For technical questions, bug reports, or feature requests, please refer to the project's issue tracking system or contact the development team at sparkinCreations.

**Happy coding! 🚀**

```
element.addEventListener('touchend', handleTouchEnd, { passive: false });  
});  
}
```

```
// Problem: Modal not closing properly  
function ensureModalCleanup() {  
  // Force close all modals  
  const modals = document.querySelectorAll('.modal');  
  modals.forEach(modal => {  
    modal.style.display = 'none';  
    modal.classList.remove('active');  
  });  
}
```

```
...
```

```
// Reset body scroll  
document.body.style.overflow = "";
```

```
// Clear any modal backdrop  
const backdrops = document.querySelectorAll('.modal-backdrop');  
backdrops.forEach(backdrop => backdrop.remove());  
...
```

```
}
```

```
...
```

#### #### Performance Issues

```
```javascript
```

```
// Problem: Slow rendering with many tasks
```

```
function optimizeTaskRendering() {  
  const container = document.getElementById('taskList');
```

```
  
  // Use document fragment for batch updates  
  const fragment = document.createDocumentFragment();
```

```
  
  taskList.forEach(task => {  
    const taskElement = createTaskElement(task);  
    fragment.appendChild(taskElement);  
  });
```

```
  
  // Single DOM update  
  container.innerHTML = "";  
  container.appendChild(fragment);  
}
```

```
// Problem: Memory leaks from event listeners
```

```
function cleanupEventListeners() {  
  // Remove all task-specific listeners  
  const taskElements = document.querySelectorAll('.task');  
  taskElements.forEach(element => {  
    // Clone element to remove all listeners  
    const newElement = element.cloneNode(true);  
    element.parentNode.replaceChild(newElement, element);
```

```
  
    // Re-add essential listeners  
    initializeTaskElement(newElement);  
  });  
}
```

```
...
```

#### #### Debug Tools

```
```javascript
```

```
// Diagnostic functions
```

```
const Diagnostics = {  
  // Check data integrity  
  validateData() {
```

```

const storage = JSON.parse(localStorage.getItem('miniCycleStorage') || '{}');
const issues = [];

Object.entries(storage).forEach(([cycleName, cycle]) => {
  if (!cycle.tasks) {
    issues.push(`${cycleName}: Missing tasks array`);
  }

  if (cycle.tasks) {
    cycle.tasks.forEach((task, index) => {
      if (!task.id) {
        issues.push(`${cycleName}[${index}]: Missing task ID`);
      }
      if (!task.text) {
        issues.push(`${cycleName}[${index}]: Missing task text`);
      }
    });
  }
});

console.log('Data validation results:', issues.length === 0 ? 'All good!' : issues);
return issues;
},

// Performance monitoring
measurePerformance(functionName, fn) {
  const start = performance.now();
  const result = fn();
  const end = performance.now();

  console.log(`${functionName} took ${(end - start).toFixed(2)}ms`);
  return result;
},

// Memory usage (Chrome only)
getMemoryUsage() {
  if (performance.memory) {
    const memory = performance.memory;
    console.table({
      'Used JS Heap Size': `${(memory.usedJSHeapSize / 1048576).toFixed(2)} MB`,
      'Total JS Heap Size': `${(memory.totalJSHeapSize / 1048576).toFixed(2)} MB`,
      'JS Heap Size Limit': `${(memory.jsHeapSizeLimit / 1048576).toFixed(2)} MB`
    });
  } else {

```

```

        console.log('Memory API not available');
    }
},

// Export diagnostic report
exportDiagnostics() {
    const report = {
        timestamp: new Date().toISOString(),
        userAgent: navigator.userAgent,
        screenSize: `${screen.width}x${screen.height}`,
        windowSize: `${window.innerWidth}x${window.innerHeight}`,
        localStorage: {
            available: typeof(Storage) !== "undefined",
            usage: JSON.stringify(localStorage).length,
            quota: this.getStorageQuota()
        },
        dataValidation: this.validateData(),
        cycleCount: Object.keys(JSON.parse(localStorage.getItem('miniCycleStorage') ||
'{}')).length,
        activeTheme: document.body.className,
        features: {
            serviceWorker: 'serviceWorker' in navigator,
            touchSupport: 'ontouchstart' in window,
            draggable: 'draggable' in document.createElement('div')
        }
    };

    // Download as JSON
    const blob = new Blob([JSON.stringify(report, null, 2)], { type: 'application/json' });
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `minicycle-diagnostics-${Date.now()}.json`;
    a.click();
    URL.revokeObjectURL(url);
},

getStorageQuota() {
    try {
        // Estimate storage quota (modern browsers)
        if (navigator.storage && navigator.storage.estimate) {
            navigator.storage.estimate().then(estimate => {
                console.log(`Storage quota: ${((estimate.quota / 1048576).toFixed(2))} MB`);
                console.log(`Storage usage: ${((estimate.usage / 1048576).toFixed(2))} MB`);
            });
        }
    }
}

```



```

        });
    }
} catch (error) {
    console.log('Storage quota estimation not available');
}
}
};

```

```

// Expose diagnostics in development
if (typeof window !== 'undefined') {
    window.MiniCycleDiagnostics = Diagnostics;
}
...

```

### ### Recovery Procedures

```

````javascript
// Emergency data recovery
const DataRecovery = {
    // Create emergency backup
    createEmergencyBackup() {
        const allData = {
            storage: localStorage.getItem('miniCycleStorage'),
            active: localStorage.getItem('activeMiniCycle'),
            settings: localStorage.getItem('appSettings'),
            reminders: localStorage.getItem('remindersSettings'),
            theme: localStorage.getItem('selectedTheme')
        };

        const backup = JSON.stringify(allData);
        const blob = new Blob([backup], { type: 'application/json' });
        const url = URL.createObjectURL(blob);

        const a = document.createElement('a');
        a.href = url;
        a.download = `minicycle-emergency-backup-${Date.now()}.json`;
        a.click();

        URL.revokeObjectURL(url);
        showNotification('Emergency backup created', 'success');
    },

    // Restore from emergency backup
    restoreFromBackup(file) {

```

```

const reader = new FileReader();
reader.onload = (e) => {
  try {
    const backup = JSON.parse(e.target.result);

    // Restore each data type
    Object.entries(backup).forEach(([key, value]) => {
      if (value) {
        const storageKey = this.getStorageKeyMapping(key);
        localStorage.setItem(storageKey, value);
      }
    });

    showNotification('Backup restored successfully', 'success');
    setTimeout(() => location.reload(), 2000);

  } catch (error) {
    showNotification('Backup file is corrupted', 'error');
    console.error('Restore failed:', error);
  }
};

reader.readAsText(file);
},

getStorageKeyMapping(backupKey) {
  const mapping = {
    storage: 'miniCycleStorage',
    active: 'activeMiniCycle',
    settings: 'appSettings',
    reminders: 'remindersSettings',
    theme: 'selectedTheme'
  };
  return mapping[backupKey] || backupKey;
}
};
...

```

-----

## ## Conclusion

This developer documentation provides a comprehensive guide to understanding, extending, and maintaining the miniCycle application. The modular architecture and extensive

customization options make it possible to adapt the application for various use cases while maintaining the core cycling methodology that makes miniCycle unique.

For additional technical details, refer to the comprehensive technical documentation. For user-facing information, consult the user manual.

#### ### Quick Reference Links

- **User Manual**: `user-manual.html`
- **Comprehensive Documentation**: Technical specification document
- **Privacy Policy**: `privacy.html`
- **Terms of Service**: `terms.html`

#### ### Support and Contributing

For technical questions, bug reports, or feature requests, please refer to the project's issue tracking system or contact the development team at sparkinCreations.

**Happy coding! 🚀**