Final Project

Anilkumar
anilkumarpeddabayi@gmail.com

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager </title>
    <link rel="stylesheet" href="styles.css">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/lucide/0.263.1/lucide.min.css"
rel="stylesheet">
</head>
<body>
    <div class="app">
        <div class="container">
            <!-- Header -->
            <div class="header">
                <div class="header-content">
                    <div class="header-left">
                        <div class="logo">
                            <i data-lucide="cpu"></i>
                        </div>
                        <div class="header-text">
                            <h1>To Do List</h1>
                            <p class="status">
                                <i data-lucide="wifi"></i>
                                <span id="current-date"></span> • <span id="current-
time"></span> • System Online
                            </p>
                        </div>
                    </div>
                    <div class="header-streak">
                        <div class="streak-display">
                            <i data-lucide="flame" id="header-flame"></i>
                            <div class="streak-text">
                                <span id="header-streak-count">0</span>
                                <span> day streak</span>
                            </div>
                        </div>
                    </div>
                </div>
            </div>

            <!-- Alert for overdue tasks -->
            <div id="overdue-alert" class="alert hidden">
                <i data-lucide="alert-triangle"></i>
                <span id="alert-message"></span>
            </div>

            <div class="main-grid">
                <!-- Tasks Section -->
                <div class="tasks-section">
```

```
                    <div class="section-header">
                        <h2>
                            <i data-lucide="zap"></i>
                            Active Processes
                        </h2>
                        <button id="add-task-btn" class="btn-primary">
                            <i data-lucide="plus"></i>
                            Initialize Task
                        </button>
                    </div>

                    <div id="tasks-container" class="tasks-container">
                        <div id="empty-state" class="empty-state">
                            <i data-lucide="cpu"></i>
                            <h3>No active processes</h3>
                            <p>Initialize your first task to begin!</p>
                        </div>
                    </div>
                </div>

                <!-- Stats Section -->
                <div class="stats-section">
                    <!-- First Row -->
                    <div class="stats-row">
                        <!-- Daily Deadline Timer -->
                        <div class="card timer-card">
                            <div class="card-header">
                                <h3>
                                    <i data-lucide="clock"></i>
                                    Daily Deadline Timer
                                </h3>
                            </div>
                            <div class="card-content">
                                <div class="timer-container">
                                    <div class="circular-timer" id="daily-timer">
                                        <svg viewBox="0 0 100 100">
                                            <circle cx="50" cy="50" r="45" class="timer-
bg"></circle>
                                            <circle cx="50" cy="50" r="45" class="timer-
progress" id="timer-circle"></circle>
                                        </svg>
                                        <div class="timer-text">
                                            <span id="timer-value">24h 0m</span>
                                            <span class="timer-label">until midnight</span>
                                        </div>
                                    </div>
                                    <p id="timer-message" class="timer-message">No daily tasks
pending</p>
                                </div>
                            </div>
                        </div>

                        <!-- Experience Points -->
                        <div class="card xp-card">
```

```
                        <div class="card-header">
                            <h3>
                                <i data-lucide="zap"></i>
                                Experience Points
                            </h3>
                        </div>
                        <div class="card-content">
                            <div class="xp-display">
                                <div class="xp-value" id="net-points">0</div>
                                <div class="xp-breakdown">
                                    Earned: +<span id="total-points">0</span> | Penalties:
-<span id="penalties">0</span>
                                </div>
                            </div>
                            <div class="circular-chart" id="xp-chart">
                                <svg viewBox="0 0 100 100">
                                    <circle cx="50" cy="50" r="40" class="chart-
bg"></circle>
                                    <circle cx="50" cy="50" r="40" class="chart-progress"
id="xp-circle"></circle>
                                </svg>
                                <div class="chart-text">
                                    <span id="xp-chart-value">0</span>
                                    <span class="chart-max">/ 500</span>
                                </div>
                                <div class="chart-label">Total XP</div>
                            </div>
                        </div>
                    </div>
                </div>

                <!-- Second Row -->
                <div class="stats-row">
                    <!-- Daily Streak -->
                    <div class="card streak-card">
                        <div class="card-header">
                            <h3>
                                <i data-lucide="flame"></i>
                                Daily Streak
                            </h3>
                        </div>
                        <div class="card-content">
                            <div class="streak-display-main">
                                <div class="streak-icon">
                                    <i data-lucide="flame" id="main-flame"></i>
                                </div>
                                <div class="streak-count" id="streak-count">0 / 7
days</div>
                                <div class="streak-progress">
                                    <div class="streak-bar" id="streak-bar"></div>
                                </div>
                                <p class="streak-message">Keep the streak alive!</p>
                            </div>
                        </div>
```

```
                    </div>

                    <!-- System Diagnostics -->
                    <div class="card diagnostics-card">
                        <div class="card-header">
                            <h3>System Diagnostics</h3>
                        </div>
                        <div class="card-content">
                            <div class="diagnostics-grid">
                                <div class="diagnostic-item cyan">
                                    <div class="diagnostic-value" id="total-tasks">0</div>
                                    <div class="diagnostic-label">Total Processes</div>
                                </div>
                                <div class="diagnostic-item green">
                                    <div class="diagnostic-value" id="active-tasks">0</div>
                                    <div class="diagnostic-label">Active Now</div>
                                </div>
                                <div class="diagnostic-item red">
                                    <div class="diagnostic-value" id="overdue-
tasks">0</div>
                                    <div class="diagnostic-label">Critical Alerts</div>
                                </div>
                                <div class="diagnostic-item yellow">
                                    <div class="diagnostic-value" id="penalty-
points">0</div>
                                    <div class="diagnostic-label">Penalty Points</div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>

                <!-- Third Row - System Analysis -->
                <div class="stats-row full-width">
                    <div class="card analysis-card">
                        <div class="card-header">
                            <h3>
                                <i data-lucide="trending-up"></i>
                                System Analysis
                            </h3>
                        </div>
                        <div class="card-content">
                            <div class="analysis-charts">
                                <div class="circular-chart large" id="completed-chart">
                                    <svg viewBox="0 0 100 100">
                                        <circle cx="50" cy="50" r="40" class="chart-
bg"></circle>
                                        <circle cx="50" cy="50" r="40" class="chart-
progress green" id="completed-circle"></circle>
                                    </svg>
                                    <div class="chart-text">
                                        <span id="completed-value">0</span>
                                        <span class="chart-max">/ <span id="completed-
max">1</span></span>
```

```
                                    </div>
                                    <div class="chart-label">Tasks Complete</div>
                                </div>
                                <div class="circular-chart large" id="success-chart">
                                    <svg viewBox="0 0 100 100">
                                        <circle cx="50" cy="50" r="40" class="chart-
bg"></circle>
                                        <circle cx="50" cy="50" r="40" class="chart-
progress cyan" id="success-circle"></circle>
                                    </svg>
                                    <div class="chart-text">
                                        <span id="success-value">0</span>
                                        <span class="chart-max">/ 100</span>
                                    </div>
                                    <div class="chart-label">Success Rate %</div>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Task Modal -->
    <div id="task-modal" class="modal hidden">
        <div class="modal-content">
            <div class="modal-header">
                <h3>Deploy New Task</h3>
                <button id="close-modal" class="close-btn">&times;</button>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="task-name">Process Name</label>
                    <input type="text" id="task-name" placeholder="Enter task identifier">
                </div>
                <div class="form-row">
                    <div class="form-group">
                        <label for="start-time">Start Time</label>
                        <input type="time" id="start-time">
                    </div>
                    <div class="form-group">
                        <label for="end-time">End Time</label>
                        <input type="time" id="end-time">
                    </div>
                </div>
                <div class="form-group">
                    <label for="points">Reward Points</label>
                    <input type="number" id="points" value="10" min="1" max="100">
                </div>
                <div class="form-group">
                    <label for="frequency">Frequency</label>
                    <select id="frequency">
```

```
                        <option value="today">Today Only</option>
                        <option value="daily">Daily</option>
                        <option value="weekly">Weekly</option>
                        <option value="monthly">Monthly</option>
                    </select>
                </div>
                <div class="form-group">
                    <label for="category">Category</label>
                    <select id="category">
                        <option value="study">📚 Study</option>
                        <option value="work">💼 Work</option>
                        <option value="gym">💪 Gym</option>
                        <option value="skill">🎯 Skill</option>
                        <option value="personal">👤 Personal</option>
                        <option value="health">🏥 Health</option>
                        <option value="other">📋 Other</option>
                    </select>
                </div>
                <div class="warning">
                    ⚠ WARNING: Incomplete tasks will result in 50% point penalty
                </div>
                <div class="modal-actions">
                    <button id="deploy-task" class="btn-success">Deploy Task</button>
                    <button id="sync-calendar" class="btn-outline">
                        <i data-lucide="calendar"></i>
                        Sync Calendar
                    </button>
                </div>
            </div>
        </div>
    </div>

    <script src="https://unpkg.com/lucide@latest/dist/umd/lucide.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Css

```
/* Reset and Base Styles */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
    background: linear-gradient(135deg, #1f2937 0%, #1e3a8a 50%, #581c87 100%);
    min-height: 100vh;
    color: white;
    overflow-x: hidden;
```

```
}

.app {
    min-height: 100vh;
    padding: 1rem;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
}

/* Header Styles */
.header {
    margin-bottom: 2rem;
}

.header-content {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 1rem;
}

.header-left {
    display: flex;
    align-items: center;
    gap: 0.75rem;
}

.logo {
    padding: 0.5rem;
    background: rgba(59, 130, 246, 0.2);
    border-radius: 0.5rem;
}

.logo i {
    width: 2rem;
    height: 2rem;
    color: #60a5fa;
}

.header-text h1 {
    font-size: 2.5rem;
    font-weight: bold;
    margin-bottom: 0.25rem;
}

.status {
    color: #93c5fd;
    font-size: 1.125rem;
    display: flex;
    align-items: center;
    gap: 0.5rem;
```

```css
}

.status i {
    width: 1rem;
    height: 1rem;
}

.header-streak {
    display: flex;
    align-items: center;
    gap: 0.75rem;
    background: rgba(31, 41, 55, 0.5);
    border-radius: 0.5rem;
    padding: 0.75rem;
    border: 1px solid rgba(251, 146, 60, 0.3);
}

.streak-display {
    display: flex;
    align-items: center;
    gap: 0.5rem;
}

.streak-display i {
    width: 1.25rem;
    height: 1.25rem;
    color: #60a5fa;
    filter: drop-shadow(0 0 4px rgba(59, 130, 246, 0.6));
}

.streak-text span:first-child {
    color: #fb923c;
    font-weight: 600;
}

.streak-text span:last-child {
    color: #9ca3af;
}

/* Alert Styles */
.alert {
    background: rgba(127, 29, 29, 0.2);
    border: 1px solid rgba(239, 68, 68, 0.3);
    color: #fca5a5;
    padding: 1rem;
    border-radius: 0.5rem;
    margin-bottom: 1.5rem;
    display: flex;
    align-items: center;
    gap: 0.5rem;
}

.alert i {
    width: 1rem;
```

```
    height: 1rem;
}

.hidden {
    display: none !important;
}

/* Main Grid */
.main-grid {
    display: grid;
    grid-template-columns: 2fr 1fr;
    gap: 1.5rem;
}

@media (max-width: 1024px) {
    .main-grid {
        grid-template-columns: 1fr;
    }
}

/* Tasks Section */
.tasks-section {
    display: flex;
    flex-direction: column;
    gap: 1rem;
}

.section-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.section-header h2 {
    font-size: 1.5rem;
    font-weight: 600;
    display: flex;
    align-items: center;
    gap: 0.5rem;
}

.section-header i {
    width: 1.5rem;
    height: 1.5rem;
    color: #fbbf24;
}

.btn-primary {
    background: linear-gradient(to right, #2563eb, #7c3aed);
    color: white;
    border: none;
    padding: 0.5rem 1rem;
    border-radius: 0.375rem;
    cursor: pointer;
```

```
    display: flex;
    align-items: center;
    gap: 0.5rem;
    font-weight: 500;
    transition: all 0.2s;
}

.btn-primary:hover {
    background: linear-gradient(to right, #1d4ed8, #6d28d9);
}

.btn-primary i {
    width: 1rem;
    height: 1rem;
}

/* Tasks Container */
.tasks-container {
    display: flex;
    flex-direction: column;
    gap: 0.75rem;
}

.empty-state {
    padding: 2rem;
    text-align: center;
    background: rgba(31, 41, 55, 0.5);
    border: 1px solid #374151;
    border-radius: 0.5rem;
}

.empty-state i {
    width: 3rem;
    height: 3rem;
    color: #9ca3af;
    margin-bottom: 0.5rem;
}

.empty-state h3 {
    font-size: 1.125rem;
    font-weight: 500;
    color: #d1d5db;
    margin-bottom: 0.5rem;
}

.empty-state p {
    color: #6b7280;
}

/* Task Card */
.task-card {
    background: rgba(31, 41, 55, 0.5);
    border: 1px solid #374151;
    border-left: 4px solid;
```

```
    border-radius: 0.5rem;
    padding: 1rem;
}

.task-card.completed {
    border-left-color: #10b981;
    background: rgba(6, 78, 59, 0.2);
    color: #34d399;
}

.task-card.active {
    border-left-color: #06b6d4;
    background: rgba(8, 145, 178, 0.2);
    color: #22d3ee;
}

.task-card.overdue {
    border-left-color: #ef4444;
    background: rgba(127, 29, 29, 0.2);
    color: #f87171;
}

.task-card.pending {
    border-left-color: #6b7280;
    color: #9ca3af;
}

.task-content {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.task-main {
    display: flex;
    align-items: center;
    gap: 0.75rem;
    flex: 1;
}

.task-checkbox {
    width: 1.25rem;
    height: 1.25rem;
    accent-color: #10b981;
}

.task-details {
    flex: 1;
}

.task-name {
    font-weight: 500;
    margin-bottom: 0.25rem;
}
```

```css
.task-name.completed {
    text-decoration: line-through;
    color: #6b7280;
}

.task-meta {
    display: flex;
    align-items: center;
    gap: 1rem;
    flex-wrap: wrap;
}

.task-time {
    display: flex;
    align-items: center;
    gap: 0.25rem;
    font-size: 0.875rem;
    color: #9ca3af;
}

.task-time i {
    width: 1rem;
    height: 1rem;
}

.badge {
    padding: 0.125rem 0.5rem;
    border-radius: 0.25rem;
    font-size: 0.75rem;
    font-weight: 500;
}

.badge.points {
    background: rgba(30, 58, 138, 0.3);
    color: #93c5fd;
}

.badge.frequency {
    background: rgba(88, 28, 135, 0.3);
    color: #c084fc;
    border: 1px solid rgba(147, 51, 234, 0.3);
}

.badge.category {
    background: rgba(67, 56, 202, 0.3);
    color: #a5b4fc;
    border: 1px solid rgba(99, 102, 241, 0.3);
}

.badge.penalty {
    background: rgba(127, 29, 29, 0.3);
    color: #fca5a5;
}
```

```css
.badge.status {
    border: 1px solid;
    background: transparent;
}

.badge.status.completed {
    border-color: rgba(16, 185, 129, 0.3);
    color: #34d399;
}

.badge.status.active {
    border-color: rgba(6, 182, 212, 0.3);
    color: #22d3ee;
}

.badge.status.overdue {
    border-color: rgba(239, 68, 68, 0.3);
    color: #f87171;
}

.badge.status.pending {
    border-color: rgba(107, 114, 128, 0.3);
    color: #9ca3af;
}

.delete-btn {
    background: none;
    border: none;
    color: #f87171;
    cursor: pointer;
    padding: 0.25rem;
    border-radius: 0.25rem;
    font-size: 1.25rem;
    transition: all 0.2s;
}

.delete-btn:hover {
    background: rgba(127, 29, 29, 0.2);
    color: #fca5a5;
}

/* Stats Section */
.stats-section {
    display: flex;
    flex-direction: column;
    gap: 1.5rem;
}

.stats-row {
    display: flex;
    flex-direction: column;
    gap: 1.5rem;
}
```

```css
.stats-row.full-width {
    display: flex;
    flex-direction: column;
    gap: 1.5rem;
}

@media (max-width: 768px) {
    .stats-row {
        gap: 1rem;
    }
}

/* Card Styles */
.card {
    background: rgba(17, 24, 39, 0.9);
    backdrop-filter: blur(12px);
    border-radius: 0.75rem;
    border: 1px solid rgba(255, 255, 255, 0.1);
    min-height: 280px;
    display: flex;
    flex-direction: column;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
    transition: all 0.3s ease;
}

.card:hover {
    transform: translateY(-2px);
    box-shadow: 0 12px 40px rgba(0, 0, 0, 0.4);
}

.timer-card {
    border-color: rgba(6, 182, 212, 0.4);
}

.xp-card {
    border-color: rgba(16, 185, 129, 0.4);
}

.streak-card {
    border-color: rgba(251, 146, 60, 0.4);
}

.diagnostics-card {
    border-color: rgba(147, 51, 234, 0.4);
}

.analysis-card {
    border-color: rgba(59, 130, 246, 0.4);
}

.card-header {
    padding: 1.25rem 1.25rem 0.75rem;
    border-bottom: 1px solid rgba(255, 255, 255, 0.05);
```

```css
}

.card-header h3 {
    font-size: 1.25rem;
    font-weight: 600;
    display: flex;
    align-items: center;
    gap: 0.75rem;
    letter-spacing: 0.025em;
}

.timer-card .card-header h3 {
    color: #22d3ee;
}

.xp-card .card-header h3 {
    color: #34d399;
}

.streak-card .card-header h3 {
    color: #fb923c;
}

.diagnostics-card .card-header h3 {
    color: #a855f7;
}

.analysis-card .card-header h3 {
    color: #60a5fa;
}

.card-header i {
    width: 1.5rem;
    height: 1.5rem;
}

.card-content {
    padding: 1.25rem;
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: center;
    overflow: visible;
}

.xp-card .card-content {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    height: auto;
    min-height: calc(100% - 3rem);
    padding-bottom: 1.5rem;
    overflow: visible;
}
```

```css
/* Timer Styles */
.timer-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100%;
}

.circular-timer {
    position: relative;
    width: 120px;
    height: 120px;
    margin-bottom: 1rem;
}

.circular-timer svg {
    width: 100%;
    height: 100%;
    transform: rotate(-90deg);
}

.timer-bg {
    fill: none;
    stroke: rgba(75, 85, 99, 0.2);
    stroke-width: 6;
}

.timer-progress {
    fill: none;
    stroke: #22d3ee;
    stroke-width: 6;
    stroke-dasharray: 283;
    stroke-dashoffset: 283;
    transition: all 1s ease-in-out;
    filter: drop-shadow(0 0 8px rgba(34, 211, 238, 0.6));
}

.timer-text {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-align: center;
}

.timer-text span:first-child {
    display: block;
    font-size: 1.125rem;
    font-weight: bold;
    color: #22d3ee;
    margin-bottom: 0.25rem;
}
```

```css
.timer-label {
    font-size: 0.875rem;
    color: #9ca3af;
}

.timer-message {
    margin-top: 0.75rem;
    font-size: 1rem;
    color: #d1d5db;
    text-align: center;
    font-weight: 500;
}

/* XP Styles */
.xp-display {
    text-align: center;
    margin-bottom: 1.5rem;
}

.xp-value {
    font-size: 2.25rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
    color: #34d399;
}

.xp-breakdown {
    font-size: 1rem;
    color: #6ee7b7;
    font-weight: 500;
}

.circular-chart {
    position: relative;
    width: 100px;
    height: 100px;
    margin: 0 auto;
}

.circular-chart.large {
    width: 140px;
    height: 140px;
}

.circular-chart svg {
    width: 100%;
    height: 100%;
    transform: rotate(-90deg);
}

.chart-bg {
    fill: none;
    stroke: rgba(75, 85, 99, 0.2);
```

```
    stroke-width: 5;
}

.chart-progress {
    fill: none;
    stroke-width: 5;
    stroke-dasharray: 251;
    stroke-dashoffset: 251;
    transition: all 1s ease-in-out;
}

.chart-progress.yellow {
    stroke: #fbbf24;
    filter: drop-shadow(0 0 6px rgba(251, 191, 36, 0.6));
}

.chart-progress.green {
    stroke: #10b981;
    filter: drop-shadow(0 0 6px rgba(16, 185, 129, 0.6));
}

.chart-progress.cyan {
    stroke: #22d3ee;
    filter: drop-shadow(0 0 6px rgba(34, 211, 238, 0.6));
}

.chart-text {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-align: center;
}

.chart-text span:first-child {
    display: block;
    font-size: 1.75rem;
    font-weight: bold;
    color: white;
}

.chart-max {
    font-size: 1rem;
    color: #9ca3af;
}

.chart-label {
    position: absolute;
    top: 100%;
    left: 50%;
    transform: translateX(-50%);
    margin-top: 1rem;
    font-size: 1rem;
    color: #d1d5db;
```

```
    text-align: center;
    white-space: nowrap;
    font-weight: 500;
}

/* Streak Styles */
.streak-display-main {
    text-align: center;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100%;
}

.streak-icon {
    margin-bottom: 1rem;
}

.streak-icon i {
    width: 2rem;
    height: 2rem;
    color: #fb923c;
    filter: drop-shadow(0 0 8px rgba(251, 146, 60, 0.6));
}

.streak-count {
    font-size: 1.75rem;
    font-weight: bold;
    margin-bottom: 1rem;
    color: #fb923c;
}

.streak-progress {
    width: 100%;
    height: 0.75rem;
    background: rgba(55, 65, 81, 0.5);
    border-radius: 9999px;
    margin-bottom: 1rem;
    overflow: hidden;
    border: 1px solid rgba(255, 255, 255, 0.1);
}

.streak-bar {
    height: 100%;
    background: linear-gradient(to right, #fb923c, #f97316);
    border-radius: 9999px;
    transition: width 0.5s ease-in-out;
    width: 0%;
    box-shadow: 0 0 8px rgba(251, 146, 60, 0.4);
}

.streak-message {
    font-size: 0.875rem;
```

```
    color: #d1d5db;
    font-weight: 500;
}

/* Diagnostics Styles */
.diagnostics-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 1rem;
    height: 100%;
}

.diagnostic-item {
    background: rgba(31, 41, 55, 0.6);
    border-radius: 0.5rem;
    padding: 1.25rem;
    border: 1px solid rgba(255, 255, 255, 0.1);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    text-align: center;
    transition: all 0.3s ease;
}

.diagnostic-item:hover {
    background: rgba(31, 41, 55, 0.8);
    transform: translateY(-1px);
}

.diagnostic-item.cyan {
    border-color: rgba(6, 182, 212, 0.3);
}

.diagnostic-item.green {
    border-color: rgba(16, 185, 129, 0.3);
}

.diagnostic-item.red {
    border-color: rgba(239, 68, 68, 0.3);
}

.diagnostic-item.yellow {
    border-color: rgba(251, 191, 36, 0.3);
}

.diagnostic-value {
    font-size: 1.75rem;
    font-weight: bold;
    margin-bottom: 0.5rem;
}

.diagnostic-item.cyan .diagnostic-value {
    color: #22d3ee;
```

```css
}

.diagnostic-item.green .diagnostic-value {
    color: #34d399;
}

.diagnostic-item.red .diagnostic-value {
    color: #f87171;
}

.diagnostic-item.yellow .diagnostic-value {
    color: #fbbf24;
}

.diagnostic-label {
    font-size: 0.875rem;
    font-weight: 500;
}

.diagnostic-item.cyan .diagnostic-label {
    color: #67e8f9;
}

.diagnostic-item.green .diagnostic-label {
    color: #6ee7b7;
}

.diagnostic-item.red .diagnostic-label {
    color: #fca5a5;
}

.diagnostic-item.yellow .diagnostic-label {
    color: #fcd34d;
}

/* Analysis Styles */
.analysis-charts {
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 2.5rem;
    height: 100%;
}

@media (max-width: 768px) {
    .analysis-charts {
        flex-direction: column;
        gap: 2rem;
    }
}

/* Card-specific content alignment */
.timer-card .card-content {
    justify-content: center;
```

```
    align-items: center;
}

.streak-card .card-content {
    justify-content: center;
    align-items: center;
}

.diagnostics-card .card-content {
    justify-content: center;
}

.analysis-card .card-content {
    justify-content: center;
    align-items: center;
}

/* Modal Styles */
.modal {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.5);
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 1000;
}

.modal-content {
    background: #1f2937;
    border: 1px solid #374151;
    border-radius: 0.5rem;
    width: 90%;
    max-width: 500px;
    max-height: 90vh;
    overflow-y: auto;
}

.modal-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    border-bottom: 1px solid #374151;
}

.modal-header h3 {
    color: #93c5fd;
    font-size: 1.125rem;
    font-weight: 500;
}
```

```css
.close-btn {
    background: none;
    border: none;
    color: #9ca3af;
    cursor: pointer;
    font-size: 1.5rem;
    padding: 0.25rem;
}

.close-btn:hover {
    color: #f3f4f6;
}

.modal-body {
    padding: 1rem;
}

.form-group {
    margin-bottom: 1rem;
}

.form-row {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 1rem;
}

.form-group label {
    display: block;
    margin-bottom: 0.25rem;
    color: #d1d5db;
    font-size: 0.875rem;
}

.form-group input,
.form-group select {
    width: 100%;
    padding: 0.5rem 0.75rem;
    background: #374151;
    border: 1px solid #4b5563;
    border-radius: 0.375rem;
    color: white;
    font-size: 0.875rem;
}

.form-group input:focus,
.form-group select:focus {
    outline: none;
    border-color: #3b82f6;
    box-shadow: 0 0 0 2px rgba(59, 130, 246, 0.2);
}

.warning {
```

```
    background: rgba(146, 64, 14, 0.2);
    border: 1px solid rgba(251, 191, 36, 0.3);
    color: #fcd34d;
    padding: 0.5rem;
    border-radius: 0.375rem;
    font-size: 0.75rem;
    margin-bottom: 1rem;
}

.modal-actions {
    display: flex;
    gap: 0.5rem;
}

.btn-success {
    flex: 1;
    background: #059669;
    color: white;
    border: none;
    padding: 0.5rem 1rem;
    border-radius: 0.375rem;
    cursor: pointer;
    font-weight: 500;
    transition: background 0.2s;
}

.btn-success:hover {
    background: #047857;
}

.btn-outline {
    flex: 1;
    background: transparent;
    color: #d1d5db;
    border: 1px solid #4b5563;
    padding: 0.5rem 1rem;
    border-radius: 0.375rem;
    cursor: pointer;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 0.5rem;
    transition: all 0.2s;
}

.btn-outline:hover {
    background: #374151;
}

.btn-outline i {
    width: 1rem;
    height: 1rem;
}
```

```css
/* Responsive Design */
@media (max-width: 768px) {
    .header-content {
        flex-direction: column;
        gap: 1rem;
    }

    .form-row {
        grid-template-columns: 1fr;
    }

    .modal-actions {
        flex-direction: column;
    }
}

/* Animations */
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

.task-card {
    animation: fadeIn 0.3s ease-out;
}

/* Glow Effects */
.glow-text {
    text-shadow: 0 0 10px rgba(34, 211, 238, 0.5);
}
```

Js

```javascript
// Task Manager Application
class TaskManager {
    constructor() {
        this.tasks = [];
        this.penalties = 0;
        this.completionDates = [];
        this.currentStreak = 0;

        this.init();
        this.loadData();
        this.setupEventListeners();
```

```
        this.startTimers();
        this.updateDateTime();
    }

    init() {
        // Initialize Lucide icons
        if (typeof lucide !== 'undefined') {
            lucide.createIcons();
        }
    }

    loadData() {
        // Load tasks from localStorage
        const savedTasks = localStorage.getItem('tasks');
        if (savedTasks) {
            this.tasks = JSON.parse(savedTasks).map(task => ({
                frequency: 'today',
                category: 'other',
                ...task,
                createdAt: new Date(task.createdAt)
            }));
        }

        // Load penalties
        const savedPenalties = localStorage.getItem('penalties');
        if (savedPenalties) {
            this.penalties = parseInt(savedPenalties);
        }

        // Load completion dates
        const savedDates = localStorage.getItem('completionDates');
        if (savedDates) {
            this.completionDates = JSON.parse(savedDates);
        }

        this.calculateStreak();
        this.updateUI();
    }

    saveData() {
        localStorage.setItem('tasks', JSON.stringify(this.tasks));
        localStorage.setItem('penalties', this.penalties.toString());
        localStorage.setItem('completionDates', JSON.stringify(this.completionDates));
    }

    setupEventListeners() {
        // Add task button
        document.getElementById('add-task-btn').addEventListener('click', () => {
            this.showModal();
        });

        // Modal close
        document.getElementById('close-modal').addEventListener('click', () => {
            this.hideModal();
```

```
        });

        // Deploy task
        document.getElementById('deploy-task').addEventListener('click', () => {
            this.addTask();
        });

        // Modal backdrop click
        document.getElementById('task-modal').addEventListener('click', (e) => {
            if (e.target.id === 'task-modal') {
                this.hideModal();
            }
        });

        // Sync calendar (placeholder)
        document.getElementById('sync-calendar').addEventListener('click', () => {
            alert('Calendar sync feature coming soon!');
        });
    }

    showModal() {
        document.getElementById('task-modal').classList.remove('hidden');
        // Reset form
        document.getElementById('task-name').value = '';
        document.getElementById('start-time').value = '';
        document.getElementById('end-time').value = '';
        document.getElementById('points').value = '10';
        document.getElementById('frequency').value = 'today';
        document.getElementById('category').value = 'other';
    }

    hideModal() {
        document.getElementById('task-modal').classList.add('hidden');
    }

    addTask() {
        const name = document.getElementById('task-name').value.trim();
        const startTime = document.getElementById('start-time').value;
        const endTime = document.getElementById('end-time').value;
        const points = parseInt(document.getElementById('points').value) || 10;
        const frequency = document.getElementById('frequency').value;
        const category = document.getElementById('category').value;

        if (!name || !startTime || !endTime) {
            alert('Please fill in all required fields');
            return;
        }

        const task = {
            id: Date.now().toString(),
            name,
            startTime,
            endTime,
            points,
```

```
            completed: false,
            createdAt: new Date(),
            penaltyApplied: false,
            frequency,
            category
        };

        this.tasks.push(task);
        this.saveData();
        this.updateUI();
        this.hideModal();
    }

    toggleTask(id) {
        const task = this.tasks.find(t => t.id === id);
        if (task) {
            task.completed = !task.completed;

            // Update completion dates for streak tracking
            if (task.completed) {
                const today = new Date().toDateString();
                if (!this.completionDates.includes(today)) {
                    this.completionDates.push(today);
                }
            }

            this.calculateStreak();
            this.saveData();
            this.updateUI();
        }
    }

    deleteTask(id) {
        this.tasks = this.tasks.filter(t => t.id !== id);
        this.saveData();
        this.updateUI();
    }

    getTaskStatus(task) {
        if (task.completed) return 'completed';

        const now = new Date();
        const currentTime = now.getHours() * 60 + now.getMinutes();
        const startTime = this.timeToMinutes(task.startTime);
        const endTime = this.timeToMinutes(task.endTime);

        if (currentTime > endTime) return 'overdue';
        if (currentTime >= startTime && currentTime <= endTime) return 'active';
        return 'pending';
    }

    timeToMinutes(timeString) {
        const [hours, minutes] = timeString.split(':').map(Number);
        return hours * 60 + minutes;
```

```
    }

    formatTime(timeString) {
        const [hours, minutes] = timeString.split(':');
        const hour = parseInt(hours);
        const ampm = hour >= 12 ? 'PM' : 'AM';
        const hour12 = hour % 12 || 12;
        return `${hour12}:${minutes} ${ampm}`;
    }

    calculateStreak() {
        const today = new Date();
        let streak = 0;

        for (let i = 0; i < 30; i++) {
            const checkDate = new Date(today);
            checkDate.setDate(today.getDate() - i);
            const dateString = checkDate.toDateString();

            if (this.completionDates.includes(dateString)) {
                streak++;
            } else {
                break;
            }
        }

        this.currentStreak = streak;
    }

    updateDateTime() {
        const now = new Date();
        const dateOptions = {
            weekday: 'long',
            year: 'numeric',
            month: 'long',
            day: 'numeric'
        };
        const timeOptions = {
            hour: '2-digit',
            minute: '2-digit',
            hour12: true
        };

        document.getElementById('current-date').textContent = now.toLocaleDateString('en-
US', dateOptions);
        document.getElementById('current-time').textContent = now.toLocaleTimeString('en-
US', timeOptions);
    }

    updateUI() {
        this.renderTasks();
        this.updateStats();
        this.updateTimer();
        this.updateStreak();
```

```
        this.updateOverdueAlert();
    }

    renderTasks() {
        const container = document.getElementById('tasks-container');
        const emptyState = document.getElementById('empty-state');

        if (this.tasks.length === 0) {
            emptyState.classList.remove('hidden');
            // Remove any existing task cards
            const taskCards = container.querySelectorAll('.task-card');
            taskCards.forEach(card => card.remove());
            return;
        }

        emptyState.classList.add('hidden');

        // Remove existing task cards
        const taskCards = container.querySelectorAll('.task-card');
        taskCards.forEach(card => card.remove());

        // Render tasks
        this.tasks.forEach(task => {
            const taskElement = this.createTaskElement(task);
            container.appendChild(taskElement);
        });
    }

    createTaskElement(task) {
        const status = this.getTaskStatus(task);
        const div = document.createElement('div');
        div.className = `task-card ${status}`;

        const categoryIcons = {
            study: '📚',
            work: '💼',
            gym: '💪',
            skill: '🎯',
            personal: '👤',
            health: '🏥',
            other: '📋'
        };

        div.innerHTML = `
            <div class="task-content">
                <div class="task-main">
                    <input type="checkbox" class="task-checkbox" ${task.completed ?
'checked' : ''}
                           onchange="taskManager.toggleTask('${task.id}')">
                    <div class="task-details">
                        <div class="task-name ${task.completed ? 'completed' :
''}">${task.name}</div>
                        <div class="task-meta">
                            <div class="task-time">
```

```
                                <i data-lucide="clock"></i>
                                ${this.formatTime(task.startTime)} -
${this.formatTime(task.endTime)}
                            </div>
                            <span class="badge points">+${task.points} pts</span>
                            <span class="badge
frequency">${task.frequency.charAt(0).toUpperCase() + task.frequency.slice(1)}</span>
                            <span class="badge category">${categoryIcons[task.category]}
${task.category.charAt(0).toUpperCase() + task.category.slice(1)}</span>
                            ${task.penaltyApplied ? `<span class="badge penalty">-
${Math.floor(task.points / 2)} penalty</span>` : ''}
                            <span class="badge status ${status}">${status}</span>
                        </div>
                    </div>
                </div>
                <button class="delete-btn"
onclick="taskManager.deleteTask('${task.id}')">&times;</button>
            </div>
        `;

        // Re-initialize Lucide icons for the new element
        if (typeof lucide !== 'undefined') {
            lucide.createIcons();
        }

        return div;
    }

    updateStats() {
        const completedTasks = this.tasks.filter(task => task.completed).length;
        const totalPoints = this.tasks.filter(task => task.completed).reduce((sum, task)
=> sum + task.points, 0);
        const netPoints = totalPoints - this.penalties;
        const completionRate = this.tasks.length > 0 ? Math.round((completedTasks /
this.tasks.length) * 100) : 0;
        const overdueTasks = this.tasks.filter(task => this.getTaskStatus(task) ===
'overdue' && !task.completed).length;
        const activeTasks = this.tasks.filter(task => this.getTaskStatus(task) ===
'active').length;

        // Update XP display
        document.getElementById('net-points').textContent = netPoints;
        document.getElementById('total-points').textContent = totalPoints;
        document.getElementById('penalties').textContent = this.penalties;

        // Update XP chart
        const xpMax = Math.max(netPoints + 100, 500);
        const xpPercentage = Math.max(0, netPoints) / xpMax * 100;
        this.updateCircularChart('xp-circle', xpPercentage);
        document.getElementById('xp-chart-value').textContent = Math.max(0, netPoints);

        // Update diagnostics
        document.getElementById('total-tasks').textContent = this.tasks.length;
        document.getElementById('active-tasks').textContent = activeTasks;
```

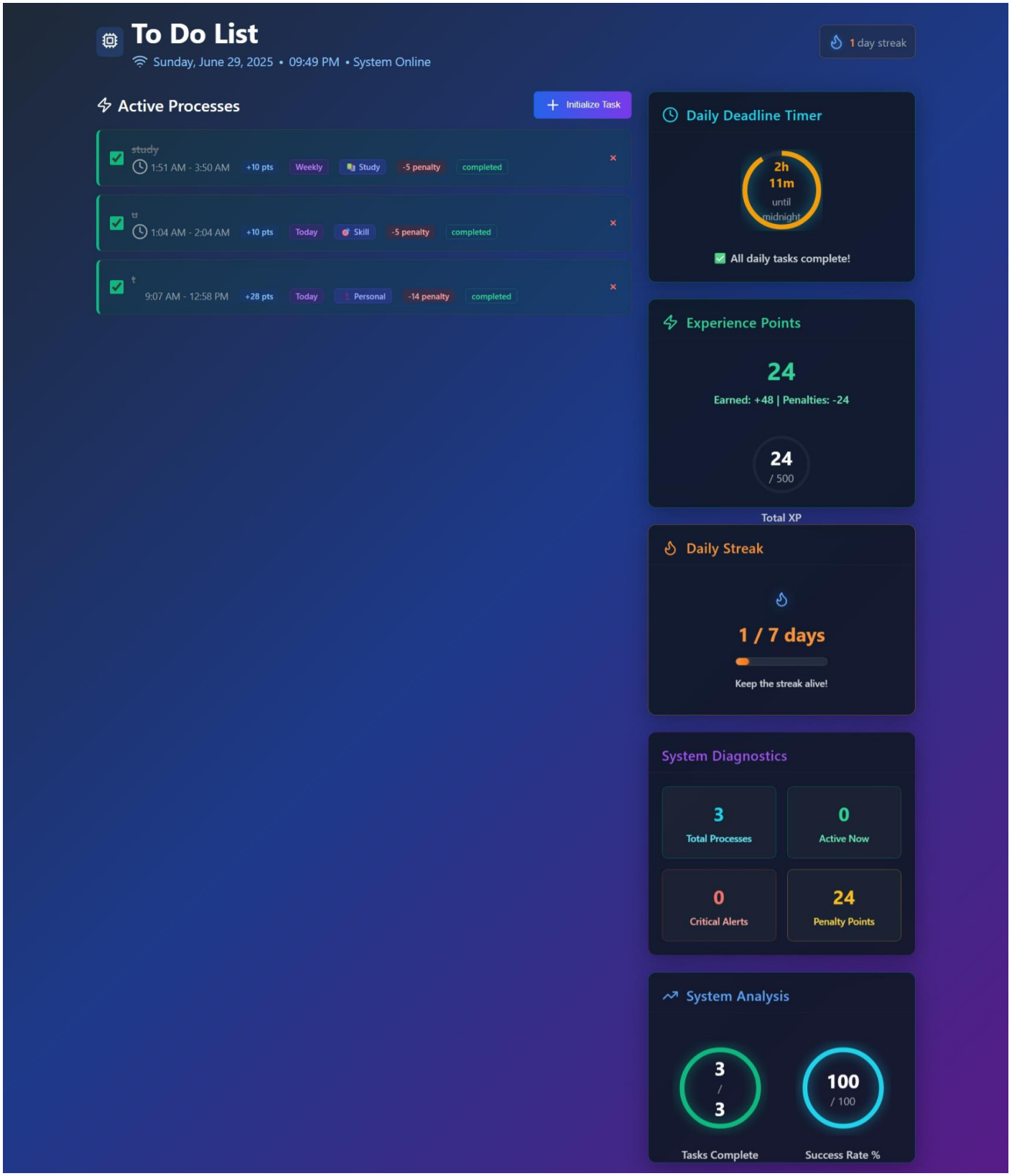
```
        document.getElementById('overdue-tasks').textContent = overdueTasks;
        document.getElementById('penalty-points').textContent = this.penalties;

        // Update analysis charts
        const completedPercentage = this.tasks.length > 0 ? (completedTasks /
this.tasks.length) * 100 : 0;
        this.updateCircularChart('completed-circle', completedPercentage);
        this.updateCircularChart('success-circle', completionRate);

        document.getElementById('completed-value').textContent = completedTasks;
        document.getElementById('completed-max').textContent = this.tasks.length || 1;
        document.getElementById('success-value').textContent = completionRate;
    }

    updateCircularChart(elementId, percentage) {
        const circle = document.getElementById(elementId);
        if (circle) {
            const circumference = 2 * Math.PI * 40; // radius = 40
            const offset = circumference - (percentage / 100) * circumference;
            circle.style.strokeDashoffset = offset;
        }
    }

    updateTimer() {
        const now = new Date();
        const endOfDay = new Date();
        endOfDay.setHours(23, 59, 59, 999);

        const totalDayMinutes = 24 * 60;
        const currentMinutes = now.getHours() * 60 + now.getMinutes();
        const remainingMinutes = Math.max(0, totalDayMinutes - currentMinutes);

        const hours = Math.floor(remainingMinutes / 60);
        const minutes = remainingMinutes % 60;

        // Update timer display
        const timerValue = document.getElementById('timer-value');
        const timerCircle = document.getElementById('timer-circle');
        const timerMessage = document.getElementById('timer-message');

        if (timerValue) {
            timerValue.textContent = hours > 0 ? `${hours}h ${minutes}m` : `${minutes}m`;
        }

        // Update timer circle
        const progress = (currentMinutes / totalDayMinutes) * 100;
        const circumference = 2 * Math.PI * 45; // radius = 45
        const offset = circumference - (progress / 100) * circumference;
        if (timerCircle) {
            timerCircle.style.strokeDashoffset = offset;

            // Update color based on time left
            if (remainingMinutes <= 60) {
                timerCircle.style.stroke = '#ef4444'; // Red
```

```
                timerValue.style.color = '#ef4444';
            } else if (remainingMinutes <= 180) {
                timerCircle.style.stroke = '#f59e0b'; // Orange
                timerValue.style.color = '#f59e0b';
            } else {
                timerCircle.style.stroke = '#22d3ee'; // Cyan
                timerValue.style.color = '#22d3ee';
            }
        }

        // Update timer message
        const incompleteDailyTasks = this.tasks.filter(
            task => (task.frequency === 'daily' || task.frequency === 'today') &&
!task.completed
        ).length;

        if (timerMessage) {
            if (incompleteDailyTasks === 0) {
                timerMessage.textContent = '☑ All daily tasks complete!';
            } else if (remainingMinutes <= 60) {
                timerMessage.textContent = '⚠ CRITICAL: Penalties incoming!';
            } else if (remainingMinutes <= 180) {
                timerMessage.textContent = '⚠ WARNING: Day ending soon!';
            } else {
                timerMessage.textContent = `${incompleteDailyTasks} daily tasks pending`;
            }
        }
    }

    updateStreak() {
        // Update header streak
        const headerStreakCount = document.getElementById('header-streak-count');
        const headerFlame = document.getElementById('header-flame');

        if (headerStreakCount) {
            headerStreakCount.textContent = this.currentStreak;
        }

        if (headerFlame) {
            if (this.currentStreak > 0) {
                headerFlame.style.color = '#60a5fa';
                headerFlame.style.filter = 'drop-shadow(0 0 4px rgba(59, 130, 246, 0.6))';
            } else {
                headerFlame.style.color = '#6b7280';
                headerFlame.style.filter = 'none';
            }
        }

        // Update main streak display
        const streakCount = document.getElementById('streak-count');
        const mainFlame = document.getElementById('main-flame');
        const streakBar = document.getElementById('streak-bar');

        if (streakCount) {
```

```
            streakCount.textContent = `${this.currentStreak} / 7 days`;
        }

        if (mainFlame) {
            if (this.currentStreak > 0) {
                mainFlame.style.color = '#60a5fa';
                mainFlame.style.filter = 'drop-shadow(0 0 8px rgba(59, 130, 246, 0.6))';
            } else {
                mainFlame.style.color = '#6b7280';
                mainFlame.style.filter = 'none';
            }
        }

        if (streakBar) {
            const streakPercentage = Math.min((this.currentStreak / 7) * 100, 100);
            streakBar.style.width = `${streakPercentage}%`;
        }
    }

    updateOverdueAlert() {
        const alert = document.getElementById('overdue-alert');
        const alertMessage = document.getElementById('alert-message');
        const overdueTasks = this.tasks.filter(task => this.getTaskStatus(task) ===
'overdue' && !task.completed);

        if (overdueTasks.length > 0) {
            const penaltyPoints = overdueTasks.reduce((sum, task) => {
                return sum + (task.penaltyApplied ? Math.floor(task.points / 2) : 0);
            }, 0);

            alertMessage.textContent = `⚠ SYSTEM ALERT: ${overdueTasks.length} task(s)
overdue! Penalty applied: -${penaltyPoints} points`;
            alert.classList.remove('hidden');
        } else {
            alert.classList.add('hidden');
        }
    }

    checkForPenalties() {
        let newPenalties = 0;
        this.tasks.forEach(task => {
            if (!task.completed && !task.penaltyApplied && this.getTaskStatus(task) ===
'overdue') {
                newPenalties += Math.floor(task.points / 2);
                task.penaltyApplied = true;
            }
        });

        if (newPenalties > 0) {
            this.penalties += newPenalties;
            this.saveData();
            this.updateUI();
        }
    }
```

```
    startTimers() {
        // Update time every second
        setInterval(() => {
            this.updateDateTime();
            this.updateTimer();
        }, 1000);

        // Check for penalties every 30 seconds
        setInterval(() => {
            this.checkForPenalties();
        }, 30000);

        // Update UI every minute
        setInterval(() => {
            this.updateUI();
        }, 60000);
    }
}

// Initialize the application when the page loads
document.addEventListener('DOMContentLoaded', () => {
    window.taskManager = new TaskManager();
});

// Handle page visibility change to update when user returns
document.addEventListener('visibilitychange', () => {
    if (!document.hidden && window.taskManager) {
        window.taskManager.updateUI();
    }
});
```

Output:-


To Do List
Sunday, June 29, 2025 • 09:49 PM • System Online
1 day streak
Active Processes
Initialize Task
study
1:51 AM - 3:50 AM
+10 pts
Weekly
Study
-5 penalty
completed
tr
1:04 AM - 2:04 AM
+10 pts
Today
Skill
-5 penalty
completed
t
9:07 AM - 12:58 PM
+28 pts
Today
Personal
-14 penalty
completed
Daily Deadline Timer
2h
11m
until
midnight
All daily tasks complete!
Experience Points
24
Earned: +48 | Penalties: -24
24
/ 500
Total XP
Daily Streak
1 / 7 days
Keep the streak alive!
System Diagnostics
3
Total Processes
0
Active Now
0
Critical Alerts
24
Penalty Points
System Analysis
3
/
3
Tasks Complete
100
/ 100
Success Rate %