exCal
Calendar Extension

Requirements Specification & Test Cases Document

Version 1.67 | December 2025

1. Executive Summary

The exCal Calendar Extension is a browser extension that intelligently detects dates and event information on web pages and enables users to add them to their calendar with minimal effort. The system leverages AI (via n8n workflow automation and OpenAI) to parse natural language dates and extract structured calendar data.

2. Project Overview

2.1 Architecture Components

The project consists of three main components:

- **Browser Extension (Chrome):** Frontend UI built with HTML/CSS/JS that captures user selections and page context
- **n8n Workflow Backend:** Webhook-based API that processes requests, interfaces with AI, and generates calendar links
- **Docker Infrastructure:** Containerized n8n deployment for consistent environment

2.2 Data Flow

1. User highlights text on a webpage or opens the extension popup
2. Extension captures: selected text, page title, URL, and page content (up to 5000 chars)
3. Data is sent to n8n webhook endpoint via POST request
4. AI Agent parses the data and extracts calendar fields (title, date, time, location, description)
5. Backend generates Google Calendar link and ICS file content
6. User reviews, edits if needed, and adds to calendar

3. Functional Requirements

3.1 Frontend Requirements (FE)

ID	Requirement	Description
FE-01	State Persistence	Store current state (selected text, AI response, user edits) in localStorage. Restore state when popup reopens within 5-minute window.
FE-02	Context Awareness	Send JSON payload with: selectedText, pageTitle, url, pageContent (limited to 5000 chars for token

ID	Requirement	Description
		efficiency).
FE-03	Review UI	Display editable form fields (title, date, time, description) before calendar submission. Allow user corrections.
FE-04	Multi-Export	Support both Google Calendar link (opens in new tab) and ICS file download for universal calendar compatibility.
FE-05	Error Handling	Display user-friendly error messages. Show loading state during API calls. Handle network failures gracefully.
FE-06	Keyboard Shortcut	Support Ctrl+Shift+E (Windows) / Cmd+Shift+E (Mac) to open extension popup.

3.2 Backend Requirements (BE)

ID	Requirement	Description
BE-01	Context Priority Logic	Prioritize selectedText as primary source. Fall back to pageContent for missing info. Handle empty selection by analyzing full page.
BE-02	AI Prompt Engineering	Optimized system prompt to minimize hallucinations. Include current date context. Request JSON-only output.
BE-03	Error Handling	Implement retry mechanism for invalid AI responses. Return standardized error codes to frontend.
BE-04	Link Generation	Generate properly encoded Google Calendar URLs. Create valid ICS file content with correct timezone handling.
BE-05	Data Persistence	Store generated calendar objects in n8n DataTable for analytics and debugging.
BE-06	Date Inference	Infer year if missing (assume next occurrence). Default to 1-hour duration if unspecified.

4. Non-Functional Requirements

ID	Requirement	Description
NFR-01	Security	Implement API key authentication for webhook endpoints. Ensure stateless execution (no data leakage between users).
NFR-02	Performance	API response time < 5 seconds (including AI processing). Extension popup loads within 500ms.
NFR-03	Scalability	Architecture supports horizontal scaling via load balancer + multiple n8n workers.
NFR-04	Reliability	System handles AI service unavailability gracefully. Retry logic with exponential backoff.
NFR-05	Compatibility	Chrome browser support (Manifest V3). Cross-calendar support (Google, Outlook, Apple via ICS).

5. Test Cases

5.1 Date Format Parsing Tests

These test cases verify the AI's ability to parse various date formats correctly.

TC-ID	Input	Expected Output	Priority
TC-001	Meeting on December 25, 2024 at 3pm	2024-12-25, 15:00	High
TC-002	Party next Tuesday at 7:30 PM	Correct next Tuesday, 19:30	High
TC-003	Conference 15.01.2025 14:00-16:00	2025-01-15, 14:00-16:00	High
TC-004	Dinner in 3 days at 8pm	Current date + 3, 20:00	Medium
TC-005	Termin am 5. März um 10 Uhr	2025-03-05, 10:00	High
TC-006	Call tomorrow morning	Tomorrow, 09:00 (default)	Medium
TC-007	2024/12/31 23:59	2024-12-31, 23:59	Medium
TC-008	Jan 5th @ noon	2025-01-05, 12:00	Medium
TC-009	Nächsten Montag 14:30	Next Monday, 14:30	High
TC-010	12/25/24 3:00 PM	2024-12-25, 15:00	Medium

5.2 Edge Case Tests

TC-ID	Scenario	Expected Behavior	Priority
TC-011	Empty page with no text content	Display error: 'No content found on page'	High
TC-012	Page with 50,000+ characters	Truncate to 5000 chars, process successfully	High
TC-013	Conflicting dates in selection	Use first/most prominent date found	Medium
TC-014	No date found in text	Display error: 'Could not find date'	High
TC-015	Network timeout to n8n	Show retry option, display timeout error	High
TC-016	Invalid JSON from AI	Retry once, then return defaults	Medium
TC-017	Special characters in title	Properly encode for URL/ICS output	Medium
TC-018	Past date selected	Allow past dates (historical events)	Low
TC-019	Popup closed mid-request	Restore state on reopen, show result	High
TC-020	Multiple events in selection	Extract primary/first event	Medium

5.3 UI/UX Tests

TC-ID	Scenario	Expected Behavior	Priority
TC-021	Open popup with no selection	Analyze full page content as fallback	Medium
TC-022	Edit fields before submission	Changes saved to localStorage immediately	High
TC-023	Click 'Add to Calendar'	Opens Google Cal link + downloads ICS	High
TC-024	Loading state visible	Spinner + 'Analyzing selection...' shown	Medium
TC-025	Success state after add	Show checkmark + 'Event Added!' message	Medium

6. Improvement Recommendations

6.1 Code Quality Improvements

- **TypeScript Migration:** Convert popup.js and config.js to TypeScript for type safety and better IDE support
- **Environment Variables:** Move API_URL from config.js to environment-specific configuration or extension storage
- **Error Boundaries:** Implement comprehensive try-catch blocks with specific error types
- **Code Documentation:** Add JSDoc comments to all functions for better maintainability

6.2 Security Enhancements

- **API Authentication:** Implement API key header authentication (currently commented out in code)
- **Input Sanitization:** Sanitize pageContent before sending to prevent injection attacks
- **HTTPS Only:** Enforce HTTPS for production webhook URL
- **Rate Limiting:** Implement rate limiting on the n8n webhook to prevent abuse

6.3 Feature Enhancements

- **Timezone Support:** Detect user timezone and include in calendar events
- **Recurring Events:** Support detection of recurring event patterns ('every Monday')
- **Calendar Integration:** Direct Google Calendar API integration (OAuth) for one-click add
- **Multi-Event Detection:** Allow batch creation of multiple events from a single page

7. Scaling & User Management

7.1 Current Architecture Limitations

- Single n8n Docker container handles all requests (single point of failure)
- No user authentication or session management
- LocalStorage persistence is device-specific, not user-specific
- OpenAI API calls have rate limits that could bottleneck under load

7.2 Recommended Scaling Architecture

Phase 1: Vertical Scaling (1-1000 users)

- Increase Docker container resources (CPU/RAM)
- Implement caching for repeated similar requests
- Add request queuing to handle burst traffic

Phase 2: Horizontal Scaling (1000-10000 users)

- Deploy load balancer (Nginx/Traefik) in front of n8n
- Run multiple n8n worker containers (stateless mode)
- Enable n8n Queue Mode with Redis for job distribution
- Separate database for persistent storage (PostgreSQL)

Phase 3: Enterprise Scale (10000+ users)

- Kubernetes deployment with auto-scaling pods

- CDN for static extension assets
- Dedicated AI inference endpoints (or multiple API keys)
- Geographic distribution for latency optimization

7.3 User Management Implementation

Authentication Options:

- OAuth 2.0 with Google (leverage existing calendar integration)
- Anonymous usage with device fingerprinting for free tier
- API key system for power users/developers

User Data Model:

- User profile (email, preferences, timezone)
- Usage quotas (events created per day/month)
- Event history (for 'recent events' feature)
- Subscription tier (free/premium)

7.4 Monitoring & Observability

- Implement request tracing (OpenTelemetry)
- Dashboard for key metrics: requests/sec, error rate, AI latency
- Alerting for anomalies (spike in errors, high latency)
- Log aggregation for debugging (ELK stack or similar)