Rhine Water Level Monitor - Technical Specification

Module Specifications

1. API Module (`js/api.js`)

Interface

```
class RheinPegelAPI {
  constructor(apiUrl)
  async fetchCurrentLevel()
  parseXMLResponse(xmlString)
  convertGermanDecimal(germanNumber)
}
```

Methods

`fetchCurrentLevel()`

Returns: `Promise<WaterLevelData>`

```
{
  waterLevel: 368,          // in cm, converted from German decimal
  date: "27. Oktober 2025",
  time: "15:25",
  timestamp: 1698415500000,
  graphic: "pegel_4.jpg"
}
```

Error Handling:

- Network timeout: 10 seconds
- Retry logic: 3 attempts with exponential backoff
- CORS fallback: Use proxy if direct fetch fails

`parseXMLResponse(xmlString)`

Input: XML string from API
Output: Parsed object
Logic:

```
const parser = new DOMParser();
const xmlDoc = parser.parseFromString(xmlString, 'text/xml');
const datum = xmlDoc.querySelector('Datum')?.textContent;
const uhrzeit = xmlDoc.querySelector('Uhrzeit')?.textContent;
const pegel = xmlDoc.querySelector('Pegel')?.textContent;
```

`convertGermanDecimal(germanNumber)`

Input: "3,68" (German format with comma)
Output: 368 (converted to cm as integer)
Logic:

```
const meters = parseFloat(germanNumber.replace(',', '.'));
return Math.round(meters * 100); // Convert to cm
```

2. Storage Module (`js/storage.js`)

Interface

```
class WaterLevelStorage {
  constructor(storageKey)
  saveReading(data)
  getHistoricalData(hours = 24)
  clearOldData(maxAge = 86400000)
  exportData()
}
```

Storage Schema

```
{
  "rhein-pegel-history": {
    "readings": [
      {
        "timestamp": 1698415500000,
        "waterLevel": 368,
        "date": "27. Oktober 2025",
        "time": "15:25"
      }
    ],
    "lastUpdated": 1698415500000,
    "version": "1.0.0"
  }
}
```

Data Retention

- Maximum entries: 1440 (one per minute for 24 hours)
- Auto-cleanup: Remove entries older than 24 hours
- Storage limit: ~2MB maximum

3. Chart Module (`js/chart.js`)

Interface

```
class WaterLevelChart {
  constructor(canvasElement)
  initialize(historicalData)
  updateChart(newData)
  addThresholdLines()
  destroy()
}
```

Chart.js Configuration

```
{
  type: 'line',
  data: {
    labels: [], // timestamps
    datasets: [{
      label: 'Wasserstand (cm)',
      data: [],
      borderColor: '#2196F3',
      backgroundColor: 'rgba(33, 150, 243, 0.1)',
      tension: 0.4,
      fill: true
    }, {
      // Warning threshold line at 400cm
      label: 'Warnstufe',
      data: Array(24).fill(400),
      borderColor: '#FF9800',
      borderDash: [5, 5],
      borderWidth: 2,
      pointRadius: 0
    }, {
      // Danger threshold line at 800cm
      label: 'Gefahrstufe',
      data: Array(24).fill(800),
      borderColor: '#F44336',
      borderDash: [5, 5],
      borderWidth: 2,
      pointRadius: 0
    }]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
      legend: {
        display: true,
        position: 'top'
      },
      tooltip: {
        mode: 'index',
        intersect: false
      }
    },
    scales: {
      x: {
        type: 'time',
        time: {
          unit: 'hour',
          displayFormats: {
            hour: 'HH:mm'
          }
        },
        title: {
          display: true,
          text: 'Zeit'
        }
      },
      y: {
        beginAtZero: true,
        title: {
          display: true,
          text: 'Wasserstand (cm)'
        },
        ticks: {
          callback: function(value) {
            return value + ' cm';
          }
        }
      }
    }
  }
}
```

4. Alert System

Alert Level Calculation

```
const ALERT_LEVELS = {
  NORMAL: {
    min: 0,
    max: 400,
    color: '#4CAF50',
    bgColor: 'rgba(76, 175, 80, 0.1)',
    label: 'Normal',
    labelDE: 'Normal',
    icon: '✓',
    description: 'Der Wasserstand liegt im normalen Bereich.'
  },
  WARNING: {
    min: 400,
    max: 800,
    color: '#FF9800',
    bgColor: 'rgba(255, 152, 0, 0.1)',
    label: 'Warning',
    labelDE: 'Warnung',
    icon: '⚠',
    description: 'Erhöhter Wasserstand - Vorsicht geboten.'
  },
  DANGER: {
    min: 800,
    max: Infinity,
    color: '#F44336',
    bgColor: 'rgba(244, 67, 54, 0.1)',
    label: 'Danger',
    labelDE: 'Gefahr',
    icon: '⚡',
    description: 'Hochwassergefahr - Extreme Vorsicht!'
  }
};

function getAlertLevel(waterLevel) {
  if (waterLevel < 400) return ALERT_LEVELS.NORMAL;
  if (waterLevel < 800) return ALERT_LEVELS.WARNING;
  return ALERT_LEVELS.DANGER;
}
```

5. Main Application (`js/app.js`)

Application State

```
const AppState = {
  currentLevel: null,
  lastUpdate: null,
  isLoading: false,
  hasError: false,
  errorMessage: null,
  autoRefreshEnabled: true,
  refreshInterval: 60000, // 60 seconds
  refreshTimer: null
};
```

Application Lifecycle

```
class RheinPegelApp {
  constructor() {
    this.api = new RheinPegelAPI('https://www.stadt-koeln.de/interne-
dienste/hochwasser/pegel_ws.php');
    this.storage = new WaterLevelStorage('rhein-pegel-history');
    this.chart = null;
  }

  async initialize() {
    // 1. Load historical data
    const history = this.storage.getHistoricalData(24);

    // 2. Initialize chart
    const canvas = document.getElementById('waterLevelChart');
    this.chart = new WaterLevelChart(canvas);
    this.chart.initialize(history);

    // 3. Fetch current data
    await this.fetchAndUpdate();

    // 4. Start auto-refresh
    this.startAutoRefresh();

    // 5. Setup event listeners
    this.setupEventListeners();
  }

  async fetchAndUpdate() {
    try {
      this.setLoading(true);
      const data = await this.api.fetchCurrentLevel();

      // Save to storage
      this.storage.saveReading(data);

      // Update UI
      this.updateDisplay(data);

      // Update chart
      this.chart.updateChart(data);

      // Update state
      AppState.currentLevel = data.waterLevel;
      AppState.lastUpdate = data.timestamp;
      AppState.hasError = false;

    } catch (error) {
      this.handleError(error);
    } finally {
      this.setLoading(false);
    }
  }

  updateDisplay(data) {
    // Update water level display
    const levelElement = document.getElementById('currentLevel');
    levelElement.textContent = data.waterLevel;

    // Update timestamp
    const timeElement = document.getElementById('lastUpdate');
    timeElement.textContent = `${data.date} ${data.time}`;

    // Update alert status
    const alertLevel = getAlertLevel(data.waterLevel);
    this.updateAlertStatus(alertLevel);
  }

  updateAlertStatus(alertLevel) {
    const statusCard = document.getElementById('statusCard');
    const statusBadge = document.getElementById('statusBadge');
    const statusIcon = document.getElementById('statusIcon');
    const statusText = document.getElementById('statusText');

    // Update colors
    statusCard.style.backgroundColor = alertLevel.bgColor;
    statusCard.style.borderColor = alertLevel.color;
    statusBadge.style.backgroundColor = alertLevel.color;

    // Update text
    statusIcon.textContent = alertLevel.icon;
    statusText.textContent = alertLevel.labelDE;

    // Update description
    const description = document.getElementById('statusDescription');
    description.textContent = alertLevel.description;
  }

  startAutoRefresh() {
    if (AppState.refreshTimer) {
      clearInterval(AppState.refreshTimer);
    }

    AppState.refreshTimer = setInterval(() => {
      if (AppState.autoRefreshEnabled) {
        this.fetchAndUpdate();
      }
    }, AppState.refreshInterval);
  }

  setupEventListeners() {
    // Manual refresh button
    document.getElementById('refreshBtn').addEventListener('click', () => {
      this.fetchAndUpdate();
    });

    // Auto-refresh toggle
    document.getElementById('autoRefreshToggle').addEventListener('change', (e)
=> {
      AppState.autoRefreshEnabled = e.target.checked;
    });
  }
}
```

HTML Structure

Main Layout

```
<!DOCTYPE html>
<html lang="de">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Rhein Pegel Köln - Echtzeit Wasserstand</title>
  <link rel="stylesheet" href="css/main.css">
  <link rel="stylesheet" href="css/responsive.css">
</head>
<body>
  <header class="app-header">
    <div class="container">
      <h1>🌊 Rhein Pegel Köln</h1>
      <p class="subtitle">Echtzeit Wasserstand-Überwachung</p>
    </div>
  </header>

  <main class="app-main">
    <div class="container">

      <!-- Current Level Card -->
      <section id="statusCard" class="status-card">
        <div class="status-header">
          <span id="statusIcon" class="status-icon">✓</span>
          <span id="statusBadge" class="status-badge">Normal</span>
        </div>
        <div class="level-display">
          <span id="currentLevel" class="level-number">---</span>
          <span class="level-unit">cm</span>
        </div>
        <p id="statusDescription" class="status-description"></p>
        <div class="last-update">
          Letzte Aktualisierung: <span id="lastUpdate">---</span>
        </div>
      </section>
      <!-- Chart Section -->
      <section class="chart-section">
        <h2>Verlauf (24 Stunden)</h2>
        <div class="chart-container">
          <canvas id="waterLevelChart"></canvas>
        </div>
      </section>

      <!-- Legend -->
      <section class="legend-section">
        <h3>Warnstufen</h3>
        <div class="legend-items">
          <div class="legend-item">
            <span class="legend-color" style="background: #4CAF50"></span>
            <span>Normal (< 400 cm)</span>
          </div>
          <div class="legend-item">
            <span class="legend-color" style="background: #FF9800"></span>
            <span>Warnung (400-800 cm)</span>
          </div>
          <div class="legend-item">
            <span class="legend-color" style="background: #F44336"></span>
            <span>Gefahr (> 800 cm)</span>
          </div>
        </div>
      </section>

      <!-- Controls -->
      <section class="controls-section">
        <button id="refreshBtn" class="btn btn-primary">
          🔄 Aktualisieren
        </button>
        <label class="toggle-switch">
          <input type="checkbox" id="autoRefreshToggle" checked>
          <span class="slider"></span>
          Auto-Aktualisierung (60s)
        </label>
      </section>

    </div>
  </main>

  <footer class="app-footer">
    <div class="container">
      <p>Datenquelle: <a href="https://www.stadt-koeln.de"
target="_blank">Stadt Köln</a></p>
      <p class="disclaimer">Diese Daten dienen nur zur Information. Für
offizielle Hochwasserwarnungen beachten Sie bitte die offiziellen Stellen.</p>
    </div>
  </footer>

  <!-- Loading Overlay -->
  <div id="loadingOverlay" class="loading-overlay hidden">
    <div class="spinner"></div>
  </div>

  <!-- Error Toast -->
  <div id="errorToast" class="toast toast-error hidden">
    <span id="errorMessage"></span>
    <button class="toast-close">&times;</button>
  </div>

  <!-- Scripts -->
  <script
src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js">
</script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-
fns@3.0.0/dist/chartjs-adapter-date-fns.bundle.min.js"></script>
  <script src="js/storage.js"></script>
  <script src="js/api.js"></script>
  <script src="js/chart.js"></script>
  <script src="js/app.js"></script>
  <script>
    // Initialize app
    document.addEventListener('DOMContentLoaded', () => {
      const app = new RheinPegelApp();
      app.initialize();
    });
  </script>
</body>
</html>
```

CSS Architecture

Main Styles (`css/main.css`)

```
:root {
  /* Colors */
  --color-primary: #2196F3;
  --color-normal: #4CAF50;
  --color-warning: #FF9800;
  --color-danger: #F44336;
  --color-background: #F5F5F5;
  --color-surface: #FFFFFF;
  --color-text-primary: #212121;
  --color-text-secondary: #757575;

  /* Spacing */
  --spacing-xs: 0.5rem;
  --spacing-sm: 1rem;
  --spacing-md: 1.5rem;
  --spacing-lg: 2rem;
```

```css
  --spacing-xl: 3rem;

  /* Border Radius */
  --radius-sm: 4px;
  --radius-md: 8px;
  --radius-lg: 12px;

  /* Shadows */
  --shadow-sm: 0 2px 4px rgba(0,0,0,0.1);
  --shadow-md: 0 4px 8px rgba(0,0,0,0.15);
  --shadow-lg: 0 8px 16px rgba(0,0,0,0.2);

  /* Transitions */
  --transition-fast: 150ms ease;
  --transition-normal: 300ms ease;
  --transition-slow: 500ms ease;
}

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
Ubuntu, sans-serif;
  background-color: var(--color-background);
  color: var(--color-text-primary);
  line-height: 1.6;
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 var(--spacing-md);
}

/* Header */
.app-header {
  background: linear-gradient(135deg, #2196F3 0%, #1976D2 100%);
  color: white;
  padding: var(--spacing-lg) 0;
  box-shadow: var(--shadow-md);
}

.app-header h1 {
  font-size: 2rem;
  margin-bottom: var(--spacing-xs);
}

.subtitle {
  opacity: 0.9;
```

```css
}

/* Status Card */
.status-card {
  background: var(--color-surface);
  border-radius: var(--radius-lg);
  padding: var(--spacing-lg);
  margin: var(--spacing-lg) 0;
  box-shadow: var(--shadow-md);
  border: 3px solid transparent;
  transition: all var(--transition-normal);
}

.level-display {
  text-align: center;
  margin: var(--spacing-md) 0;
}

.level-number {
  font-size: 4rem;
  font-weight: bold;
  color: var(--color-primary);
}

.level-unit {
  font-size: 2rem;
  color: var(--color-text-secondary);
}

/* Chart */
.chart-section {
  background: var(--color-surface);
  border-radius: var(--radius-lg);
  padding: var(--spacing-lg);
  margin: var(--spacing-lg) 0;
  box-shadow: var(--shadow-md);
}

.chart-container {
  position: relative;
  height: 400px;
  margin-top: var(--spacing-md);
}

/* Responsive */
@media (max-width: 768px) {
  .level-number {
    font-size: 3rem;
  }

  .chart-container {
    height: 300px;
  }
}
```

CORS Handling Strategy

Option 1: Direct Fetch (Preferred)

```
async fetchCurrentLevel() {
  try {
    const response = await fetch(this.apiUrl, {
      method: 'GET',
      mode: 'cors',
      cache: 'no-cache'
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const xmlText = await response.text();
    return this.parseXMLResponse(xmlText);
  } catch (error) {
    // Fallback to proxy if CORS fails
    return this.fetchWithProxy();
  }
}
```

Option 2: CORS Proxy (Fallback)

```
async fetchWithProxy() {
  const proxyUrl = 'https://cors-anywhere.herokuapp.com/';
  const response = await fetch(proxyUrl + this.apiUrl);
  const xmlText = await response.text();
  return this.parseXMLResponse(xmlText);
}
```

Option 3: Local Proxy (Production)

Create a simple Node.js proxy server:

```
// server.js
const express = require('express');
const cors = require('cors');
const fetch = require('node-fetch');
const app = express();
app.use(cors());

app.get('/api/pegel', async (req, res) => {
  try {
    const response = await fetch('https://www.stadt-koeln.de/interne-
dienste/hochwasser/pegel_ws.php');
    const xml = await response.text();
    res.set('Content-Type', 'text/xml');
    res.send(xml);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.listen(3000);
```

Testing Strategy

Unit Tests

- API XML parsing
- German decimal conversion
- Alert level calculation
- Data storage/retrieval

Integration Tests

- API fetch → Storage → Display
- Chart updates with new data
- Auto-refresh mechanism

Browser Tests

- Chrome, Firefox, Safari, Edge
- Mobile browsers (iOS Safari, Chrome Mobile)
- Different screen sizes

Error Scenarios

- Network timeout
- Invalid XML response
- localStorage full
- Chart.js load failure

Performance Benchmarks

Target Metrics

- First Contentful Paint: < 1.5s
- Time to Interactive: < 2.5s
- API Response: < 500ms
- Chart Render: < 100ms
- localStorage Read/Write: < 10ms

Optimization Techniques

- Lazy load Chart.js
- Debounce resize events
- Use requestAnimationFrame for animations
- Minimize DOM manipulations
- Cache DOM references

Accessibility Checklist

- [ ] Semantic HTML5 elements
- [ ] ARIA labels for dynamic content
- [ ] Keyboard navigation (Tab, Enter, Space)
- [ ] Screen reader announcements
- [ ] Color contrast ratio > 4.5:1
- [ ] Focus indicators visible
- [ ] Alt text for graphics
- [ ] Skip navigation links
- [ ] Responsive font sizes
- [ ] Print stylesheet

Deployment Checklist

- [ ] Minify CSS and JavaScript
- [ ] Optimize images
- [ ] Add favicon
- [ ] Configure HTTPS
- [ ] Set up CDN for Chart.js
- [ ] Add meta tags (description, keywords)
- [ ] Configure robots.txt
- [ ] Add sitemap.xml
- [ ] Test on all target browsers
- [ ] Validate HTML/CSS
- [ ] Run Lighthouse audit
- [ ] Configure error tracking

Version Control Strategy

Git Workflow

```
main
  ├── develop
  │   ├── feature/api-integration
  │   ├── feature/chart-visualization
  │   └── feature/responsive-design
  └── hotfix/cors-issue
```

Commit Message Format

```
<type>(<scope>): <subject>

<body>

<footer>
```

Types: feat, fix, docs, style, refactor, test, chore

Documentation Requirements

1. **README.md**: Setup, usage, deployment
2. **ARCHITECTURE.md**: System design, decisions
3. **API_DOCS.md**: API integration details
4. **CHANGELOG.md**: Version history
5. **CONTRIBUTING.md**: Development guidelines
6. **LICENSE**: MIT or similar

This completes the technical specification for the Rhine Water Level Monitor web application.