EXECUTIVE DASHBOARD

Development Handoff Document

Generated: February 02, 2026

Project: VP of Value Brands - Executive Command Center

Version: 3.0 (With PDF Export & Bug Fixes)

Contents:

1. PROJECT OVERVIEW

Technology Stack:

- Backend: Google Apps Script (Code.gs)
- Frontend: HTML5/CSS3/JavaScript (index.html)
- Data Source: Google Sheets (Deliverables, Meetings, Alerts)
- Calendar: Google Calendar API Integration
- Styling: Custom CSS with Verizon Brand Colors (#cd040b)

Key Features:

- Executive Dashboard with Overview, Schedule, Deliverables, Documents, Links pages
- 2-Week Meeting Schedule View with PDF Export
- Conflict Detection for Overlapping Meetings
- Deliverables Tracking with Urgency Badges
- Document Repository Management
- Alert Banner System
- Calendar Sync with Executive's Google Calendar

Sheet Configuration:

- DOCUMENTS_START_ROW: 163 (Document Repository begins)
- DOCUMENTS_END_ROW: 212
- Document columns: A=Description, C=Owner, D=Link, G=Comments

2. TODO LIST - REMAINING ITEMS

HIGH PRIORITY - Verification Needed:

- ■ Deploy and test Code.gs to Google Apps Script editor
- ■ Deploy and test index.html via doGet() web app
- ■ Verify document repository populates on Overview page
- ■ Verify sidebar hides dates when ETA is null
- ■ Verify Documents tab shows correct columns (Description, Owner, Link, Comments)
- ■ Verify text-based 'Days Left' values display correctly (weekly, monthly)

PDF EXPORT - Testing Required:

- ■ Test PDF export modal opens correctly
- ■ Test meeting selection checkboxes work
- ■ Test 'Select All' / 'Deselect All' buttons
- ■ Test PDF generation with selected meetings
- ■ Verify conflict indicators appear for overlapping meetings
- ■ Verify PDF layout matches the provided template design

POTENTIAL ENHANCEMENTS:

- ■ Add loading spinner during PDF generation
- ■ Add error handling for failed API calls
- ■ Add responsive design improvements for mobile
- ■ Add dark mode support
- ■ Add filtering/search for deliverables
- ■ Add sorting options for tables

KNOWN ISSUES TO MONITOR:

- ■ Rich text URL extraction may fail for some hyperlink formats
- ■ Calendar sync may timeout for large date ranges
- ■ PDF generation may fail if meetings data is malformed

3. COMPLETED FIXES SUMMARY

Bug Fix #1: Document Repository on Overview

Problem: Document repository items were not populating on the overview page.

Solution: Added doc-repo-section to overview page that calls renderDocumentRepoList() helper function to display first 5 documents with 'View All' button.

Files Modified: index.html

Bug Fix #2: Sidebar Date Hiding

Problem: Dates were showing under items in left sidebar even when ETA was null.

Solution: Modified renderSidebarDeliverables() to check hasValidEta before displaying ETA and days badge. Condition: d.eta && d.daysRemaining !== null && typeof d.daysRemaining === 'number'

Files Modified: index.html

Bug Fix #3: Documents Tab Headers

Problem: Documents tab was missing proper headers and pulling from wrong columns.

Solution: Updated table headers to: Description, Owner, Link to Materials, Comments. Updated backend getDocumentRepoLinks() to pull comment from row[6] (Column G) instead of row[4].

Files Modified: Code.gs, index.html

Bug Fix #4: Text-Based Days Left

Problem: Text values like 'weekly' or 'monthly' in Days Left column were being miscalculated.

Solution: Added detection for text values using regex. Text values now display with neutral gray badge instead of urgency calculation.

Files Modified: index.html

4. CODE.GS (BACKEND)

Google Apps Script - 2,140 lines

Location: /mnt/user-data/outputs/Code.gs

```
/**
 * ==========================================================================
 * EXECUTIVE DASHBOARD - COMPLETE GOOGLE APPS SCRIPT
 * ==========================================================================
 * VP of Value Brands - Executive Command Center
 *
 * FEATURES:
 * - Auto-creates sheet tabs if they don't exist
 * - Syncs executive's Google Calendar to Meetings sheet
 * - Serves dashboard with real-time data
 * - Creates agenda documents
 * - Supports recurring meeting management
 * - PDF Export for weekly schedule
 *
 * Version: 3.0 (With PDF Export &amp;amp; Enhanced 2-Week View)
 * Last Updated: February 01, 2026
 * ==========================================================================
 */

// ============================================================================
// CONFIGURATION - UPDATE THESE
// ============================================================================

const CONFIG = {
  // Main Sheet IDs
  DELIVERABLES_SHEET_ID: '1ZmC-04S_OdhuoJs-XIBiOujj5rtsJTZNOiH26_0LTgk',

  // If meetings are in same workbook, use same ID:
  MEETINGS_SHEET_ID: '1ZmC-04S_OdhuoJs-XIBiOujj5rtsJTZNOiH26_0LTgk',

  // Tab/Sheet Names (will be auto-created if missing)
  MEETINGS_TAB_NAME: 'Meetings',
  DELIVERABLES_TAB_NAME: 'Document Repository',
  ARCHIVE_TAB_NAME: 'Meeting Archive', // NEW - for completed meetings

  // Calendar Settings for Auto-Sync
  EXECUTIVE_CALENDAR_ID: 'michael.sarcone@verizon.com',
  SYNC_DAYS_AHEAD: 90,      // Pull meetings 90 days forward
  SYNC_DAYS_BEHIND: 30,     // Pull meetings 30 days back

  // Row ranges for Document Repository sheet
  //
  // ■ IMPORTANT: These ranges are FLEXIBLE!
  // The code uses DYNAMIC detection and stops at the first empty row.
  // You can add/remove rows freely - just update START_ROW if sections move.
  // END_ROW values are safety limits, not hard requirements.
  //
  // ■ You CAN change section names in the sheet without breaking code
  // ■ You CAN add more rows without updating END_ROW
  // ■ You CAN move sections - just update START_ROW values below

  EXEC1_START_ROW: 3,          // Exec 1 Deliverables (Skipping Title)
  EXEC1_END_ROW: 20,

  EXEC2_START_ROW: 23,         // Exec 2 Deliverables (Skipping Title)
  EXEC2_END_ROW: 35,

  ADMIN_START_ROW: 38,         // Admin &amp;amp; Settings Links (Skipping Title)
  ADMIN_END_ROW: 45,

  KPI_START_ROW: 49,           // KPIs &amp;amp; Performance (Skipping Title)
  KPI_END_ROW: 60,

  EXEC_UPDATES_START_ROW: 63,  // Executive Updates Archive (Skipping Title)
  EXEC_UPDATES_END_ROW: 110,

  DOCUMENTS_START_ROW: 163,    // Document Repository Section (Skipping Title)
  DOCUMENTS_END_ROW: 212,

  COMPLETED_START_ROW: 215,    // Completed Items (Skipping Title)
  COMPLETED_END_ROW: 314,

  // Alert Banner Settings
  ALERT_SHEET_NAME: 'Alerts',
  ALERT_START_ROW: 2,

  // Column indices for Meetings sheet (0-indexed)
  MEETING_COLUMNS: {
    COMPLETE: 0,         // A
    APPROVED: 1,         // B
```

CODE.GS (continued - lines 81+)

```
    HOT_TOPIC: 2,        // C
    MEETING_NAME: 3,     // D
    DATE: 4,             // E
    TIME: 5,             // F
    DURATION: 6,         // G
    FREQUENCY: 7,        // H
    CATEGORY: 8,         // I
    DESCRIPTION: 9,      // J
    ATTENDEES: 10,       // K
    DELIVERABLE_LINK: 11,// L
    PREP_REQUIRED: 12,   // M
    NOTES: 14,           // O (N is empty column)
    EVENT_ID: 15         // P
  }
};

// ============================================================================
// MAIN FUNCTIONS
// ============================================================================

/**
 * Setup function - Initializes sheets (run this first!)
 */
function setup() {
  ensureSheetsExist();
  Logger.log('■ Setup complete!');
  Logger.log('■ Sheets verified/created');
  Logger.log('');
  Logger.log('■ Next steps:');
  Logger.log('1. Refresh your Google Sheet to see the menu');
  Logger.log('2. Use menu: ■ Executive Dashboard &gt; ■ Sync Calendar Now');
  Logger.log('3. Deploy as Web App (Deploy &gt; New Deployment)');
}

/**
 * Create custom menu - called automatically when sheet opens
 */
function onOpen() {
  createCustomMenu();
}

/**
 * Create custom menu in Google Sheets
 */
function createCustomMenu() {
  const ui = SpreadsheetApp.getUi();
  ui.createMenu('■ Executive Dashboard')
    .addItem('■ Sync Calendar Now', 'syncCalendarToMeetingsSheet')
    .addItem('■ Setup Daily Auto-Sync', 'setupDailySyncTrigger')
    .addItem('■■ Remove Auto-Sync', 'removeSyncTriggers')
    .addSeparator()
    .addItem('■ Archive Completed Meetings', 'archiveCompletedMeetings')
    .addSeparator()
    .addItem('■■ Initialize Sheets', 'ensureSheetsExist')
    .addItem('■ Test Data Retrieval', 'testDataRetrieval')
    .addToUi();
}

/**
 * Ensure all required sheets exist, create if missing
 */
function ensureSheetsExist() {
  const ss = SpreadsheetApp.openById(CONFIG.DELIVERABLES_SHEET_ID);

  let sheetsCreated = [];

  // Create Meetings sheet if missing
  let meetingsSheet = ss.getSheetByName(CONFIG.MEETINGS_TAB_NAME);
  if (!meetingsSheet) {
    Logger.log('■ Creating Meetings sheet...');
    meetingsSheet = ss.insertSheet(CONFIG.MEETINGS_TAB_NAME);
    initializeMeetingsSheet(meetingsSheet);
    sheetsCreated.push(CONFIG.MEETINGS_TAB_NAME);
  } else {
    Logger.log('■ Meetings sheet already exists');
  }

  // Create Deliverables sheet if missing
  let deliverablesSheet = ss.getSheetByName(CONFIG.DELIVERABLES_TAB_NAME);
  if (!deliverablesSheet) {
```

CODE.GS (continued - lines 161+)

```
    Logger.log('■ Creating Team Updates sheet...');
    deliverablesSheet = ss.insertSheet(CONFIG.DELIVERABLES_TAB_NAME);
    initializeDeliverablesSheet(deliverablesSheet);
    sheetsCreated.push(CONFIG.DELIVERABLES_TAB_NAME);
  } else {
    Logger.log('■ Team Updates sheet already exists');
  }

  // Show results
  if (sheetsCreated.length &gt; 0) {
    Logger.log('■ Created sheets: ' + sheetsCreated.join(', '));
    // Try to show UI alert if possible, but don't fail if not
    try {
      SpreadsheetApp.getUi().alert('■ Sheets Created!\n\nNew sheets:\n• ' + sheetsCreated.join('\n• '));
    } catch (e) {
      Logger.log('(Running from script editor - UI alerts disabled)');
    }
  } else {
    Logger.log('■ All required sheets already exist');
    try {
      SpreadsheetApp.getUi().alert('■ All sheets verified!\n\nExisting sheets:\n• ' + CONFIG.MEETINGS_TAB_NAME + '\n• ' + CONFIG.DELIVERABLES_TAB_NAME);
    } catch (e) {
      Logger.log('(Running from script editor - UI alerts disabled)');
    }
  }
}

/**
 * Initialize Meetings sheet with headers
 */
function initializeMeetingsSheet(sheet) {
  const headers = [
    'COMPLETE', 'APPROVED', 'HOT TOPIC', 'MEETING NAME', 'DATE', 'TIME',
    'DURATION', 'FREQUENCY', 'CATEGORY', 'DESCRIPTION', 'ATTENDEES',
    'DELIVERABLE LINK', 'PREP REQUIRED', '', 'NOTES', 'EVENT ID'
  ];

  sheet.getRange(1, 1, 1, headers.length).setValues([headers]);
  sheet.getRange(1, 1, 1, headers.length)
    .setFontWeight('bold')
    .setBackground('#1e293b')
    .setFontColor('#ffffff')
    .setHorizontalAlignment('center');

  // Set column widths
  sheet.setColumnWidth(1, 80);  // Complete
  sheet.setColumnWidth(2, 80);  // Approved
  sheet.setColumnWidth(3, 90);  // Hot Topic
  sheet.setColumnWidth(4, 250); // Meeting Name
  sheet.setColumnWidth(5, 100); // Date
  sheet.setColumnWidth(6, 80);  // Time
  sheet.setColumnWidth(7, 90);  // Duration
  sheet.setColumnWidth(8, 100); // Frequency
  sheet.setColumnWidth(9, 120); // Category
  sheet.setColumnWidth(10, 300);// Description
  sheet.setColumnWidth(11, 200);// Attendees
  sheet.setColumnWidth(12, 200);// Deliverable Link
  sheet.setColumnWidth(13, 110);// Prep Required
  sheet.setColumnWidth(14, 50); // Empty
  sheet.setColumnWidth(15, 250);// Notes
  sheet.setColumnWidth(16, 200);// Event ID

  // Format checkboxes
  sheet.getRange('A2:C1000').insertCheckboxes();
  sheet.getRange('M2:M1000').insertCheckboxes();

  // Freeze header row
  sheet.setFrozenRows(1);

  Logger.log('■ Meetings sheet initialized with headers and formatting');
}

/**
 * Initialize Deliverables sheet with section headers
 */
function initializeDeliverablesSheet(sheet) {
  const sections = [
    { row: 1, title: 'Exec 1 Upcoming Deliverables &amp;amp; Hot Topics' },
    { row: 21, title: 'Exec 2 Governance Upcoming Deliverables &amp;amp; Hot Topics' },
    { row: 36, title: 'Admin &amp;amp; Settings - Quick Links' },
```

CODE.GS (continued - lines 241+)

```
    { row: 47, title: 'KPIs &amp;amp; Performance Dashboards' },
    { row: 61, title: 'Executive Updates - Completed Hot Topics Archive' },
    { row: 161, title: 'Document Repository' },
    { row: 213, title: 'Exec 2\'s Completed Governance Items' }
  ];

  const columnHeaders = ['Description', 'ETA', 'Owner', 'Link to Materials', 'Days Left', '', 'Comment'];

  sections.forEach(section =&amp;gt; {
    // Section title
    sheet.getRange(section.row, 1, 1, 7).merge()
      .setValue(section.title)
      .setFontWeight('bold')
      .setFontSize(12)
      .setBackground('#f3f4f6')
      .setHorizontalAlignment('left');

    // Column headers
    sheet.getRange(section.row + 1, 1, 1, columnHeaders.length)
      .setValues([columnHeaders])
      .setFontWeight('bold')
      .setBackground('#e5e7eb')
      .setHorizontalAlignment('center');
  });

  // Set column widths
  sheet.setColumnWidth(1, 300); // Description
  sheet.setColumnWidth(2, 100); // ETA
  sheet.setColumnWidth(3, 120); // Owner
  sheet.setColumnWidth(4, 250); // Link
  sheet.setColumnWidth(5, 100); // Days Left
  sheet.setColumnWidth(6, 50);  // Empty
  sheet.setColumnWidth(7, 250); // Comment

  Logger.log('■ Deliverables sheet initialized with section structure');
}

// ============================================================================
// CALENDAR SYNC FUNCTIONS
// ============================================================================

/**
 * Sync executive's calendar to Meetings sheet
 */
function syncCalendarToMeetingsSheet() {
  Logger.log('■ Starting calendar sync...');

  const ss = SpreadsheetApp.openById(CONFIG.MEETINGS_SHEET_ID);
  const sheet = ss.getSheetByName(CONFIG.MEETINGS_TAB_NAME);

  if (!sheet) {
    Logger.log('■ Meetings sheet not found! Run ensureSheetsExist() first.');
    try {
      SpreadsheetApp.getUi().alert('■ Meetings sheet not found!\n\nRun "Initialize Sheets" first.');
    } catch (e) {
      Logger.log('(Run ensureSheetsExist() function to create the sheet)');
    }
    return;
  }

  // Calculate date range
  const now = new Date();
  const startDate = new Date(now.getTime() - (CONFIG.SYNC_DAYS_BEHIND * 24 * 60 * 60 * 1000));
  const endDate = new Date(now.getTime() + (CONFIG.SYNC_DAYS_AHEAD * 24 * 60 * 60 * 1000));

  Logger.log(`■ Syncing from ${startDate.toDateString()} to ${endDate.toDateString()}`);

  // Get calendar
  let calendar;
  try {
    if (CONFIG.EXECUTIVE_CALENDAR_ID === 'primary') {
      calendar = CalendarApp.getDefaultCalendar();
    } else {
      calendar = CalendarApp.getCalendarById(CONFIG.EXECUTIVE_CALENDAR_ID);
    }
  } catch (e) {
    Logger.log('■ Error accessing calendar: ' + e);
    try {
      SpreadsheetApp.getUi().alert('■ Cannot access calendar!\n\n' + e);
    } catch (uiError) {
```

CODE.GS (continued - lines 321+)

```
      Logger.log('(Run from Google Sheets to see UI alerts)');
    }
    return;
  }

  // Get events
  const events = calendar.getEvents(startDate, endDate);
  Logger.log(`■ Found ${events.length} calendar events`);

  // Get existing data to preserve manual edits
  const existingData = getExistingMeetingData(sheet);

  // Process events
  const newRows = [];
  for (const event of events) {
    const title = event.getTitle();

    // Skip all-day events
    if (event.isAllDayEvent()) {
      continue;
    }

    // Skip noisy meetings (DNS, lunch, 1:1, no attendees, no virtual link)
    if (shouldFilterMeeting(event, title)) {
      continue;
    }

    // Get event details
    const startTime = event.getStartTime();
    const endTime = event.getEndTime();
    const duration = Math.round((endTime - startTime) / (1000 * 60)); // minutes

    // Skip 0-minute meetings
    if (duration &lt;= 0) continue;

    // Get description
    const description = event.getDescription() || '';
    const location = event.getLocation() || '';

    const eventId = event.getId();
    const existingRow = existingData[eventId];

    // Get attendees (limit to first 5)
    const guests = event.getGuestList();
    const attendees = guests.slice(0, 5)
      .map(g =&gt; g.getName() || g.getEmail().split('@')[0])
      .join(', ');

    // Clean description - remove HTML formatting
    const cleanDescription = cleanHtmlDescription(description);

    // Determine meeting type and frequency
    const meetingType = determineMeetingType(title);
    const frequency = determineFrequency(event, title);

    // Build row - preserve checkbox states if meeting exists
    const row = [
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.COMPLETE] : false,        // A: Complete
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.APPROVED] : false,        // B: Approved
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.HOT_TOPIC] : false,       // C: Hot Topic
      title,                                                                       // D: Meeting Name
      startTime,                                                                   // E: Date
      Utilities.formatDate(startTime, Session.getScriptTimeZone(), 'HH:mm'),     // F: Time
      duration + ' min',                                                           // G: Duration
      frequency,                                                                   // H: Frequency
      meetingType,                                                                 // I: Category
      cleanDescription,                                                            // J: Description
      attendees,                                                                   // K: Attendees
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.DELIVERABLE_LINK] : '',   // L: Deliverable Link
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.PREP_REQUIRED] : false,   // M: Prep Required
      '',                                                                          // N: Empty column
      existingRow ? existingRow[CONFIG.MEETING_COLUMNS.NOTES] : '',              // O: Notes
      eventId                                                                      // P: Event ID
    ];

    newRows.push(row);
  }

  // Clear existing data (except header)
  if (sheet.getLastRow() &gt; 1) {
```

CODE.GS (continued - lines 401+)

```
    sheet.getRange(2, 1, sheet.getLastRow() - 1, 16).clearContent();
  }

  // Write new data
  if (newRows.length &gt; 0) {
    sheet.getRange(2, 1, newRows.length, 16).setValues(newRows);
  }

  Logger.log(`■ Sync complete! ${newRows.length} meetings synced`);
  try {
    SpreadsheetApp.getUi().alert(`■ Calendar Sync Complete!\n\n${newRows.length} meetings synced\nFrom: ${startDate.toDateString()}\nTo: ${endDate.toDat
  } catch (e) {
    Logger.log('(Sync complete - view results in the sheet)');
  }
}

/**
 * Get existing meeting data to preserve manual edits
 */
function getExistingMeetingData(sheet) {
  const data = {};
  if (sheet.getLastRow() &lt; 2) return data;

  const range = sheet.getRange(2, 1, sheet.getLastRow() - 1, 16);
  const values = range.getValues();

  values.forEach(row =&gt; {
    const eventId = row[CONFIG.MEETING_COLUMNS.EVENT_ID];
    if (eventId) {
      data[eventId] = row;
    }
  });

  return data;
}

/**
 * Determine meeting type from title
 */
function determineMeetingType(title) {
  const titleLower = title.toLowerCase();

  if (titleLower.includes('1:1') || titleLower.includes('one on one')) return '1:1';
  if (titleLower.includes('staff') || titleLower.includes('team')) return 'Staff Meeting';
  if (titleLower.includes('board') || titleLower.includes('executive')) return 'Executive';
  if (titleLower.includes('standup') || titleLower.includes('daily')) return 'Standup';
  if (titleLower.includes('review') || titleLower.includes('retrospective')) return 'Review';
  if (titleLower.includes('planning')) return 'Planning';
  if (titleLower.includes('sync')) return 'Sync';
  if (titleLower.includes('deep work') || titleLower.includes('focus')) return 'Deep Work';

  return 'Meeting';
}

/**
 * Determine frequency from event
 */
function determineFrequency(event, title) {
  const titleLower = title.toLowerCase();

  if (titleLower.includes('daily')) return 'Daily';
  if (titleLower.includes('weekly')) return 'Weekly';
  if (titleLower.includes('biweekly') || titleLower.includes('bi-weekly')) return 'Bi-weekly';
  if (titleLower.includes('monthly')) return 'Monthly';
  if (titleLower.includes('quarterly')) return 'Quarterly';

  // Check if it's a recurring event
  // Note: EventRecurrence is not directly accessible in Apps Script
  // This is a simplified check
  return 'One-time';
}

/**
 * Check if event is non-meeting (OOO, holiday, DNS, lunch, 1:1, etc.)
 */
function isNonMeetingEvent(title) {
  const titleLower = title.toLowerCase();
  const nonMeetingKeywords = [
    'ooo', 'out of office', 'vacation', 'pto', 'holiday',
    'off', 'personal', 'dentist', 'doctor', 'appointment',
```

CODE.GS (continued - lines 481+)

```
    'dns', 'do not schedule', 'lunch', 'break', 'blocked', 'hold',
    '1:1', '1-1', 'one on one', 'one-on-one', '1 on 1',
    'focus time', 'no meetings'
  ];

  return nonMeetingKeywords.some(keyword =&gt; titleLower.includes(keyword));
}

/**
 * Check if meeting should be filtered out (no attendees or no virtual link)
 */
function shouldFilterMeeting(event, title) {
  // First check title-based filters (DNS, Lunch, OOO, etc.)
  if (isNonMeetingEvent(title)) {
    return true;
  }

  // Rule for Attendees and Virtual Links has been removed.
  // All other events will now be synced to the sheet.
  return false;
}

/**
 * Clean HTML formatting from description
 */
function cleanHtmlDescription(html) {
  if (!html) return '';

  // Remove HTML tags but keep the text content
  let cleaned = html
    // Remove &lt;br&gt;, &lt;br/&gt;, &lt;br /&gt; tags with newlines
    .replace(/&lt;br\s*\/?&gt;/gi, '\n')
    // Remove HTML tags
    .replace(/&lt;[^&gt;]+&gt;/g, '')
    // Decode common HTML entities
    .replace(/&amp;nbsp;/g, ' ')
    .replace(/&amp;amp;/g, '&amp;')
    .replace(/&amp;lt;/g, '&lt;')
    .replace(/&amp;gt;/g, '&gt;')
    .replace(/&amp;quot;/g, '"')
    .replace(/&amp;#39;/g, "'")
    // Remove multiple spaces
    .replace(/\s+/g, ' ')
    // Remove multiple newlines
    .replace(/\n\s*\n/g, '\n')
    // Trim
    .trim();

  // Limit length to avoid cell overflow (max 500 chars)
  if (cleaned.length &gt; 500) {
    cleaned = cleaned.substring(0, 497) + '...';
  }

  return cleaned;
}

/**
 * Setup daily automatic calendar sync
 */
function setupDailySyncTrigger() {
  // Remove existing triggers first
  removeSyncTriggers();

  // Create new daily trigger at 6 AM
  ScriptApp.newTrigger('syncCalendarToMeetingsSheet')
    .timeBased()
    .atHour(6)
    .everyDays(1)
    .create();

  Logger.log('■ Daily sync trigger created (6 AM daily)');
  try {
    SpreadsheetApp.getUi().alert('■ Automatic Daily Sync Enabled!\n\nCalendar will sync every day at 6 AM.\n\nTo disable, run "Remove Auto-Sync" from th
  } catch (e) {
    Logger.log('(Trigger created successfully - check Triggers in Apps Script)');
  }
}

/**
 * Remove all sync triggers
```

CODE.GS (continued - lines 561+)

```
 */
function removeSyncTriggers() {
  const triggers = ScriptApp.getProjectTriggers();
  let removed = 0;
  triggers.forEach(trigger =&gt; {
    if (trigger.getHandlerFunction() === 'syncCalendarToMeetingsSheet') {
      ScriptApp.deleteTrigger(trigger);
      removed++;
    }
  });
  Logger.log(`■ Removed ${removed} sync trigger(s)`);
  try {
    if (removed &gt; 0) {
      SpreadsheetApp.getUi().alert(`■ Auto-sync disabled!\n\n${removed} trigger(s) removed.`);
    } else {
      SpreadsheetApp.getUi().alert('■■ No auto-sync triggers found.');
    }
  } catch (e) {
    Logger.log('(Triggers removed - view in Apps Script &gt; Triggers)');
  }
}

// ============================================================================
// WEB APP FUNCTIONS
// ============================================================================

/**
 * Serves the HTML dashboard
 */
function doGet(e) {
  const template = HtmlService.createTemplateFromFile('index');
  return template.evaluate()
    .setTitle('Value Governance Dashboard')
    .setFaviconUrl('https://ssl.gstatic.com/docs/spreadsheets/favicon3.ico')
    .addMetaTag('viewport', 'width=device-width, initial-scale=1.0');
}

/**
 * Include HTML files
 */
function include(filename) {
  return HtmlService.createHtmlOutputFromFile(filename).getContent();
}

// ============================================================================
// DATA RETRIEVAL FUNCTIONS
// ============================================================================

/**
 * Get all dashboard data in one call
 */
function getAllDashboardData() {
  try {
    Logger.log('Starting getAllDashboardData...');

    const result = {
      exec1Deliverables: [],
      exec2Deliverables: [],
      meetings: [],
      recurringMeetings: [],
      adminLinks: [],
      kpiLinks: [],
      executiveUpdates: [],
      documentRepository: [],
      documentRepoLinks: [],
      completedItems: []
    };

    // Get each section with individual error handling
    try {
      result.exec1Deliverables = getExec1Deliverables();
      Logger.log('■ Exec1: ' + result.exec1Deliverables.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Exec1: ' + e);
    }

    try {
      result.exec2Deliverables = getExec2Deliverables();
      Logger.log('■ Exec2: ' + result.exec2Deliverables.length + ' items');
    } catch (e) {
```

CODE.GS (continued - lines 641+)

```
      Logger.log('■ Error getting Exec2: ' + e);
    }

    try {
      result.meetings = getMeetings();
      Logger.log('■ Meetings: ' + result.meetings.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Meetings: ' + e);
      Logger.log('Full error: ' + e.stack);
    }

    try {
      result.recurringMeetings = getRecurringMeetings();
      Logger.log('■ Recurring: ' + result.recurringMeetings.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Recurring: ' + e);
    }

    try {
      result.adminLinks = getAdminLinks();
      Logger.log('■ Admin: ' + result.adminLinks.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Admin: ' + e);
    }

    try {
      result.kpiLinks = getKPILinks();
      Logger.log('■ KPI: ' + result.kpiLinks.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting KPI: ' + e);
    }

    try {
      result.executiveUpdates = getExecutiveUpdates();
      Logger.log('■ Exec Updates: ' + result.executiveUpdates.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Exec Updates: ' + e);
    }

    try {
      result.documentRepository = getDocumentRepository();
      Logger.log('■ Documents: ' + result.documentRepository.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Documents: ' + e);
    }

    try {
      result.documentRepoLinks = getDocumentRepoLinks();
      Logger.log('■ Doc Repo Links: ' + result.documentRepoLinks.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Doc Repo Links: ' + e);
    }

    try {
      result.completedItems = getCompletedItems();
      Logger.log('■ Completed: ' + result.completedItems.length + ' items');
    } catch (e) {
      Logger.log('■ Error getting Completed: ' + e);
    }

    Logger.log('Returning data...');

    // *** CRITICAL FIX: Force JSON serialization to strip out problematic Date objects ***
    const jsonSafeData = JSON.parse(JSON.stringify(result));
    return jsonSafeData;

  } catch (error) {
    Logger.log('■■■ CRITICAL ERROR in getAllDashboardData: ' + error);
    Logger.log('Stack: ' + error.stack);
    return {
      error: error.toString(),
      exec1Deliverables: [],
      exec2Deliverables: [],
      meetings: [],
      recurringMeetings: [],
      adminLinks: [],
      kpiLinks: [],
      executiveUpdates: [],
      documentRepository: [],
      documentRepoLinks: [],
```

CODE.GS (continued - lines 721+)

```
      completedItems: []
    };
  }
}

/**
 * Simple test function to verify backend connection
 */
function testConnection() {
  return {
    status: 'success',
    message: 'Backend is working!',
    timestamp: new Date().toISOString(),
    sheetId: CONFIG.DELIVERABLES_SHEET_ID
  };
}

/**
 * Test if getAllDashboardData can be serialized
 */
function testSerializability() {
  try {
    const data = getAllDashboardData();
    const jsonString = JSON.stringify(data);

    return {
      success: true,
      dataSize: jsonString.length,
      itemCounts: {
        exec1: data.exec1Deliverables.length,
        exec2: data.exec2Deliverables.length,
        meetings: data.meetings.length,
        recurring: data.recurringMeetings.length,
        admin: data.adminLinks.length,
        kpi: data.kpiLinks.length,
        updates: data.executiveUpdates.length,
        documents: data.documentRepository.length,
        completed: data.completedItems.length
      }
    };
  } catch (e) {
    return {
      success: false,
      error: e.toString(),
      stack: e.stack
    };
  }
}

/**
 * Archive completed meetings to separate sheet
 */
function archiveCompletedMeetings() {
  const sheet = SpreadsheetApp.openById(CONFIG.MEETINGS_SHEET_ID).getSheetByName('Meetings');
  let archive = SpreadsheetApp.openById(CONFIG.MEETINGS_SHEET_ID).getSheetByName('Meeting Archive');

  if (!archive) {
    archive = SpreadsheetApp.openById(CONFIG.MEETINGS_SHEET_ID).insertSheet('Meeting Archive');
    archive.getRange(1, 1, 1, 16).setValues([sheet.getRange(1, 1, 1, 16).getValues()[0]]);
    archive.getRange(1, 1, 1, 16).setFontWeight('bold').setBackground('#1e293b').setFontColor('#ffffff');
  }

  const data = sheet.getDataRange().getValues();
  const toArchive = [];
  const toDelete = [];

  for (let i = 1; i &lt; data.length; i++) {
    if (data[i][0] === true) { // COMPLETE checkbox
      toArchive.push(data[i]);
      toDelete.push(i + 1);
    }
  }

  if (toArchive.length &gt; 0) {
    const lastRow = archive.getLastRow();
    archive.getRange(lastRow + 1, 1, toArchive.length, 16).setValues(toArchive);

    for (let i = toDelete.length - 1; i &gt;= 0; i--) {
      sheet.deleteRow(toDelete[i]);
    }
```

CODE.GS (continued - lines 801+)

```
    Logger.log('■ Archived ' + toArchive.length + ' completed meetings');
    try {
      SpreadsheetApp.getUi().alert('■ Archived ' + toArchive.length + ' completed meetings');
    } catch (e) {
      Logger.log('Alert not available in this context');
    }
  } else {
    Logger.log('No completed meetings to archive');
    try {
      SpreadsheetApp.getUi().alert('No completed meetings to archive');
    } catch (e) {
      Logger.log('Alert not available in this context');
    }
  }
}

/**
 * Get Exec 1 Deliverables (DYNAMIC - stops at first empty row)
 */
function getExec1Deliverables() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.EXEC1_START_ROW;
    const maxRows = CONFIG.EXEC1_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const deliverables = [];
    const today = new Date();

    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      const eta = row[1] ? new Date(row[1]) : null;
      const daysRemaining = eta ? Math.ceil((eta - today) / (1000 * 60 * 60 * 24)) : null;

      // Extract URL from Column D hyperlink
      let linkedMaterials = '';
      const richText = richTextValues[i][3];
      if (richText) {
        linkedMaterials = richText.getLinkUrl() || '';
      }
      if (!linkedMaterials &amp;&amp; row[3]) {
        linkedMaterials = String(row[3]);
      }

      deliverables.push({
        id: 'exec1-' + i,
        description: row[0] || '',
        eta: eta ? Utilities.formatDate(eta, Session.getScriptTimeZone(), 'MMM dd') : '',
        etaFull: eta ? Utilities.formatDate(eta, Session.getScriptTimeZone(), 'yyyy-MM-dd') : '',
        owner: row[2] || '',
        linkedMaterials: linkedMaterials,
        daysRemaining: daysRemaining,
        daysRemainingText: daysRemaining !== null ? Math.abs(daysRemaining) + 'd' : '',
        comment: row[6] || '',
        isHotTopic: daysRemaining !== null &amp;&amp; daysRemaining &lt;= 7,
        isOverdue: daysRemaining !== null &amp;&amp; daysRemaining &lt; 0,
        urgency: getUrgencyLevel(daysRemaining)
      });
    }

    return deliverables;
  } catch (error) {
    Logger.log('Error getting Exec 1 deliverables: ' + error);
    return [];
  }
}

/**
 * Get Exec 2 Deliverables (DYNAMIC)
 */
function getExec2Deliverables() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.EXEC2_START_ROW;
```

CODE.GS (continued - lines 881+)

```
    const maxRows = CONFIG.EXEC2_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const deliverables = [];
    const today = new Date();

    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      const eta = row[1] ? new Date(row[1]) : null;
      const daysRemaining = eta ? Math.ceil((eta - today) / (1000 * 60 * 60 * 24)) : null;

      // Extract URL from Column D hyperlink
      let linkedMaterials = '';
      const richText = richTextValues[i][3];
      if (richText) {
        linkedMaterials = richText.getLinkUrl() || '';
      }
      if (!linkedMaterials &amp;&amp; row[3]) {
        linkedMaterials = String(row[3]);
      }

      deliverables.push({
        id: 'exec2-' + i,
        description: row[0] || '',
        eta: eta ? Utilities.formatDate(eta, Session.getScriptTimeZone(), 'MMM dd') : '',
        etaFull: eta ? Utilities.formatDate(eta, Session.getScriptTimeZone(), 'yyyy-MM-dd') : '',
        owner: row[2] || '',
        linkedMaterials: linkedMaterials,
        daysRemaining: daysRemaining,
        daysRemainingText: daysRemaining !== null ? Math.abs(daysRemaining) + 'd' : '',
        comment: row[6] || '',
        isHotTopic: daysRemaining !== null &amp;&amp; daysRemaining &lt;= 7,
        isOverdue: daysRemaining !== null &amp;&amp; daysRemaining &lt; 0,
        urgency: getUrgencyLevel(daysRemaining)
      });
    }

    return deliverables;
  } catch (error) {
    Logger.log('Error getting Exec 2 deliverables: ' + error);
    return [];
  }
}

/**
 * Get meetings for 2-week schedule view (current week + next week only)
 */
function getMeetings() {
  try {
    const sheet = getSheet(CONFIG.MEETINGS_SHEET_ID, CONFIG.MEETINGS_TAB_NAME);
    if (sheet.getLastRow() &lt; 2) return [];

    const dataRange = sheet.getDataRange();
    const values = dataRange.getValues();

    // Calculate current week's Monday and end of next week's Friday
    const today = new Date();
    today.setHours(0, 0, 0, 0);

    // Get current week's Monday
    const currentDay = today.getDay();
    const diff = currentDay === 0 ? -6 : 1 - currentDay; // Adjust to Monday
    const currentWeekMonday = new Date(today);
    currentWeekMonday.setDate(today.getDate() + diff);

    // Get end of next week's Friday
    const nextWeekFriday = new Date(currentWeekMonday);
    nextWeekFriday.setDate(currentWeekMonday.getDate() + 11); // +11 days from Monday = next Friday
    nextWeekFriday.setHours(23, 59, 59, 999);

    const meetings = [];

    for (let i = 1; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[CONFIG.MEETING_COLUMNS.MEETING_NAME]) continue;
```

CODE.GS (continued - lines 961+)

```
      // Skip completed meetings
      if (row[CONFIG.MEETING_COLUMNS.COMPLETE] === true) continue;

      // ONLY show approved meetings
      if (row[CONFIG.MEETING_COLUMNS.APPROVED] !== true) continue;

      // CLEAN DATA: Filter out noisy meetings
      const title = (row[CONFIG.MEETING_COLUMNS.MEETING_NAME] || '').toLowerCase();
      const attendees = row[CONFIG.MEETING_COLUMNS.ATTENDEES] || '';
      const link = row[CONFIG.MEETING_COLUMNS.DELIVERABLE_LINK] || '';
      const description = (row[CONFIG.MEETING_COLUMNS.DESCRIPTION] || '').toLowerCase();

      // 1. Remove meetings with no attendees AND no meeting link
      const hasLink = link.includes('http') || description.includes('webex') || description.includes('zoom') || description.includes('meet.google');
      if (!attendees &amp;&amp; !hasLink) continue;

      // 2. Remove lunch, DNS, and 1:1s
      if (title.includes('lunch') || title.includes('dns') || title.includes('1:1')) continue;

      const meetingDate = new Date(row[CONFIG.MEETING_COLUMNS.DATE]);
      const dayOfWeek = meetingDate.getDay();

      // Skip weekends (0 = Sunday, 6 = Saturday)
      if (dayOfWeek === 0 || dayOfWeek === 6) continue;

      // Only include meetings from current week Monday to next week Friday
      if (meetingDate &gt;= currentWeekMonday &amp;&amp; meetingDate &lt;= nextWeekFriday) {
        meetings.push({
          id: row[CONFIG.MEETING_COLUMNS.EVENT_ID] || 'meeting-' + i,
          title: row[CONFIG.MEETING_COLUMNS.MEETING_NAME] || '',
          date: Utilities.formatDate(meetingDate, Session.getScriptTimeZone(), 'yyyy-MM-dd'),
          dateFormatted: Utilities.formatDate(meetingDate, Session.getScriptTimeZone(), 'MMM dd'),
          time: (function(t) {
            if (!t) return '';
            let timeStr = String(t);
            // If it's a Date object from Sheets, format it
            if (t instanceof Date) {
              return Utilities.formatDate(t, Session.getScriptTimeZone(), 'h:mm a');
            }
            // If it's a string like "14:00", convert to 12h
            if (/^\d{1,2}:\d{2}/.test(timeStr)) {
              let [h, m] = timeStr.split(':');
              h = parseInt(h);
              let ampm = h &gt;= 12 ? 'PM' : 'AM';
              h = h % 12;
              h = h ? h : 12;
              return h + ':' + m.substring(0,2) + ' ' + ampm;
            }
            return timeStr;
          })(row[CONFIG.MEETING_COLUMNS.TIME]),
          time24: (function(t) {
            if (!t) return '00:00';
            if (t instanceof Date) {
              return Utilities.formatDate(t, Session.getScriptTimeZone(), 'HH:mm');
            }
            let timeStr = String(t);
            if (/^\d{1,2}:\d{2}/.test(timeStr)) {
              return timeStr.substring(0, 5);
            }
            return '00:00';
          })(row[CONFIG.MEETING_COLUMNS.TIME]),
          duration: String(row[CONFIG.MEETING_COLUMNS.DURATION] || ''),
          durationMinutes: (function(d) {
            if (!d) return 60;
            const match = String(d).match(/(\d+)/);
            return match ? parseInt(match[1]) : 60;
          })(row[CONFIG.MEETING_COLUMNS.DURATION]),
          category: row[CONFIG.MEETING_COLUMNS.CATEGORY] || 'meeting',
          description: row[CONFIG.MEETING_COLUMNS.DESCRIPTION] || '',
          attendees: row[CONFIG.MEETING_COLUMNS.ATTENDEES] || '',
          isHotTopic: row[CONFIG.MEETING_COLUMNS.HOT_TOPIC] === true,
          prepRequired: row[CONFIG.MEETING_COLUMNS.PREP_REQUIRED] === true,
          deliverableLink: row[CONFIG.MEETING_COLUMNS.DELIVERABLE_LINK] || '',
          notes: row[CONFIG.MEETING_COLUMNS.NOTES] || '',
          dayOfWeek: meetingDate.getDay(),
          weekNumber: getWeekNumber(meetingDate, currentWeekMonday)
        });
      }
    }
```

CODE.GS (continued - lines 1041+)

```
    // Sort by date and time
    meetings.sort((a, b) =&gt; {
      const dateCompare = a.date.localeCompare(b.date);
      if (dateCompare !== 0) return dateCompare;

      // Compare times
      const timeA = a.time24 || '00:00';
      const timeB = b.time24 || '00:00';
      return timeA.localeCompare(timeB);
    });

    return meetings;
  } catch (error) {
    Logger.log('Error getting meetings: ' + error);
    return [];
  }
}

/**
 * Helper function to determine which week a date falls in
 */
function getWeekNumber(date, currentWeekMonday) {
  const dateTime = date.getTime();
  const currentWeekMondayTime = currentWeekMonday.getTime();
  const nextWeekMonday = new Date(currentWeekMonday);
  nextWeekMonday.setDate(currentWeekMonday.getDate() + 7);

  if (dateTime &lt; nextWeekMonday.getTime()) {
    return 1; // Current week
  }
  return 2; // Next week
}

/**
 * Get recurring meetings for agenda builder
 */
function getRecurringMeetings() {
  try {
    const sheet = getSheet(CONFIG.MEETINGS_SHEET_ID, CONFIG.MEETINGS_TAB_NAME);
    if (sheet.getLastRow() &lt; 2) return [];

    const dataRange = sheet.getDataRange();
    const values = dataRange.getValues();

    const recurringMeetings = [];
    const seen = new Set();

    for (let i = 1; i &lt; values.length; i++) {
      const row = values[i];
      const frequency = row[CONFIG.MEETING_COLUMNS.FREQUENCY] ? row[CONFIG.MEETING_COLUMNS.FREQUENCY].toString().toLowerCase() : '';
      const meetingName = row[CONFIG.MEETING_COLUMNS.MEETING_NAME];

      if (meetingName &amp;&amp; frequency &amp;&amp;
          (frequency.includes('weekly') || frequency.includes('monthly') ||
           frequency.includes('daily') || frequency.includes('bi-weekly'))) {

        const key = meetingName.toString().trim();
        if (!seen.has(key)) {
          seen.add(key);
          recurringMeetings.push({
            id: row[CONFIG.MEETING_COLUMNS.EVENT_ID] || 'recurring-' + i,
            name: meetingName,
            frequency: row[CONFIG.MEETING_COLUMNS.FREQUENCY]
          });
        }
      }
    }

    return recurringMeetings;
  } catch (error) {
    Logger.log('Error getting recurring meetings: ' + error);
    return [];
  }
}

/**
 * Get Admin Links (DYNAMIC)
 */
function getAdminLinks() {
```

CODE.GS (continued - lines 1121+)

```
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.ADMIN_START_ROW;
    const maxRows = CONFIG.ADMIN_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const links = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      // Extract URL from Column D hyperlink
      let url = '';
      const richText = richTextValues[i][3];
      if (richText) {
        url = richText.getLinkUrl() || '';
      }
      if (!url &amp;&amp; row[3]) {
        url = String(row[3]);
      }

      links.push({
        id: 'admin-' + i,
        name: String(row[0] || ''),
        url: url,
        category: 'admin'
      });
    }

    return links;
  } catch (error) {
    Logger.log('Error getting admin links: ' + error);
    return [];
  }
}

/**
 * Get KPI Links (DYNAMIC)
 */
function getKPILinks() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.KPI_START_ROW;
    const maxRows = CONFIG.KPI_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const links = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      // Extract URL from Column D hyperlink
      let url = '';
      const richText = richTextValues[i][3];
      if (richText) {
        url = richText.getLinkUrl() || '';
      }
      if (!url &amp;&amp; row[3]) {
        url = String(row[3]);
      }

      links.push({
        id: 'kpi-' + i,
        name: String(row[0] || ''),
        url: url,
        category: 'kpi'
      });
    }

    return links;
  } catch (error) {
    Logger.log('Error getting KPI links: ' + error);
    return [];
  }
```

CODE.GS (continued - lines 1201+)

```
}

/**
 * Get Executive Updates (DYNAMIC)
 */
function getExecutiveUpdates() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.EXEC_UPDATES_START_ROW;
    const maxRows = CONFIG.EXEC_UPDATES_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const updates = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      // Extract URL from Column D hyperlink
      let url = '';
      const richText = richTextValues[i][3];
      if (richText) {
        url = richText.getLinkUrl() || '';
      }
      if (!url &amp;&amp; row[3]) {
        url = String(row[3]);
      }

      updates.push({
        id: 'update-' + i,
        name: String(row[0] || ''),
        url: url,
        dateAdded: row[1] ? Utilities.formatDate(new Date(row[1]), Session.getScriptTimeZone(), 'MMM dd, yyyy') : '',
        category: 'executive-update'
      });
    }

    return updates;
  } catch (error) {
    Logger.log('Error getting executive updates: ' + error);
    return [];
  }
}

/**
 * Get Document Repository items (DYNAMIC) - NEW SECTION!
 */
function getDocumentRepository() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.DOCUMENTS_START_ROW;
    const maxRows = CONFIG.DOCUMENTS_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();

    const documents = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      // Safely handle date formatting
      let dateAdded = '';
      try {
        if (row[1] &amp;&amp; row[1] instanceof Date) {
          dateAdded = Utilities.formatDate(row[1], Session.getScriptTimeZone(), 'MMM dd, yyyy');
        } else if (row[1]) {
          dateAdded = row[1].toString();
        }
      } catch (e) {
        Logger.log('Warning: Invalid date in Document Repository row ' + (startRow + i));
      }

      documents.push({
        id: 'document-' + i,
        name: String(row[0] || ''),
        url: String(row[3] || ''),
        dateAdded: dateAdded,
```

CODE.GS (continued - lines 1281+)

```
        owner: String(row[2] || ''),
        category: 'document'
      });
    }

    return documents;
  } catch (error) {
    Logger.log('Error getting document repository: ' + error);
    return [];
  }
}

/**
 * Get Completed Items
 */
function getCompletedItems() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
    const startRow = CONFIG.COMPLETED_START_ROW;
    const maxRows = CONFIG.COMPLETED_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();

    const completed = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      completed.push({
        id: 'completed-' + i,
        description: row[0],
        eta: row[1] instanceof Date ? Utilities.formatDate(row[1], Session.getScriptTimeZone(), 'MMM dd') : row[1],
        owner: row[2],
        link: row[3],
        daysLeft: row[4],
        comment: row[6]
      });
    }
    return completed;
  } catch (error) {
    Logger.log('Error getting completed items: ' + error);
    return [];
  }
}

/**
 * Get active alerts for the scrolling banner
 */
function getAlerts() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.ALERT_SHEET_NAME);
    if (!sheet) return [];

    const data = sheet.getDataRange().getValues();
    const alerts = [];

    // Column A: Active (Checkbox), Column B: Alert Text
    for (let i = CONFIG.ALERT_START_ROW - 1; i &lt; data.length; i++) {
      if (data[i][0] === true &amp;amp;&amp;amp; data[i][1]) {
        alerts.push(data[i][1]);
      }
    }
    return alerts;
  } catch (error) {
    Logger.log('Error getting alerts: ' + error);
    return [];
  }
}

// ============================================================================
// AGENDA CREATION FUNCTIONS
// ============================================================================

/**
 * Create agenda document
 */
function createAgenda(meetingData) {
  try {
    const { meetingName, meetingDate, topics, meetingLink } = meetingData;
```

CODE.GS (continued - lines 1361+)

```
    const doc = DocumentApp.create(`${meetingName} - Agenda - ${meetingDate}`);
    const body = doc.getBody();

    // Title
    const title = body.appendParagraph(meetingName);
    title.setHeading(DocumentApp.ParagraphHeading.HEADING1);
    title.setAlignment(DocumentApp.HorizontalAlignment.CENTER);

    // Date
    const dateP = body.appendParagraph(meetingDate);
    dateP.setAlignment(DocumentApp.HorizontalAlignment.CENTER);
    body.appendParagraph('');

    // Meeting link
    if (meetingLink) {
      const linkP = body.appendParagraph('Meeting Link: ');
      linkP.appendText(meetingLink).setLinkUrl(meetingLink);
      body.appendParagraph('');
    }

    // Agenda topics
    const topicsHeading = body.appendParagraph('Agenda Topics');
    topicsHeading.setHeading(DocumentApp.ParagraphHeading.HEADING2);

    topics.forEach((topic, index) =&gt; {
      if (topic &amp;&amp; topic.trim() !== '') {
        const topicP = body.appendParagraph(`${index + 1}. ${topic}`);
        topicP.setIndentFirstLine(36);
        body.appendParagraph('   Notes:');
        body.appendParagraph('');
      }
    });

    // Action items
    body.appendParagraph('');
    const actionHeading = body.appendParagraph('Action Items');
    actionHeading.setHeading(DocumentApp.ParagraphHeading.HEADING2);
    body.appendParagraph('• ');
    body.appendParagraph('• ');
    body.appendParagraph('• ');

    // Next steps
    body.appendParagraph('');
    const nextStepsHeading = body.appendParagraph('Next Steps');
    nextStepsHeading.setHeading(DocumentApp.ParagraphHeading.HEADING2);
    body.appendParagraph('');

    doc.saveAndClose();

    return {
      success: true,
      documentUrl: doc.getUrl(),
      documentId: doc.getId(),
      documentName: doc.getName()
    };

  } catch (error) {
    Logger.log('Error creating agenda: ' + error);
    return {
      success: false,
      error: error.toString()
    };
  }
}

// ============================================================================
// PDF EXPORT FUNCTIONS
// ============================================================================

/**
 * Generate PDF of weekly schedule
 * @param {Object} options - Export options including selected meetings, week title, etc.
 * @return {Object} - Contains success status and download URL or error
 */
function generateWeeklySchedulePDF(options) {
  try {
    const { selectedMeetingIds, weekTitle, includePrepMeetings, agendaItems } = options;

    // Get all meetings
```

CODE.GS (continued - lines 1441+)

```
const allMeetings = getMeetings();

// Filter to selected meetings only
let meetings = allMeetings;
if (selectedMeetingIds &amp;amp;&amp;amp; selectedMeetingIds.length &amp;gt; 0) {
  meetings = allMeetings.filter(m =&amp;gt; selectedMeetingIds.includes(m.id));
}

// Create a new Google Doc for the PDF
const doc = DocumentApp.create('Weekly Schedule - ' + (weekTitle || new Date().toLocaleDateString()));
const body = doc.getBody();

// Set page to landscape
body.setPageWidth(792).setPageHeight(612); // Letter landscape
body.setMarginTop(36);
body.setMarginBottom(36);
body.setMarginLeft(36);
body.setMarginRight(36);

// Title
const title = body.appendParagraph(weekTitle || 'The Current Week: A Funnel from Data to Decision');
title.setHeading(DocumentApp.ParagraphHeading.HEADING1);
title.setForegroundColor('#ffffff');
title.setBackgroundColor('#000000');
title.setAlignment(DocumentApp.HorizontalAlignment.LEFT);
title.setSpacingAfter(12);

// Group meetings by date
const meetingsByDate = {};
meetings.forEach(m =&amp;gt; {
  if (!meetingsByDate[m.date]) {
    meetingsByDate[m.date] = [];
  }
  meetingsByDate[m.date].push(m);
});

// Get unique dates and sort
const dates = Object.keys(meetingsByDate).sort();

// Create table for schedule
const table = body.appendTable();

// Header row with day names and dates
const headerRow = table.appendTableRow();
const dayNames = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'];

// Calculate the Monday of the first week
let firstDate = new Date(dates[0] + 'T00:00:00');
const firstDayOfWeek = firstDate.getDay();
const mondayOffset = firstDayOfWeek === 0 ? -6 : 1 - firstDayOfWeek;
const monday = new Date(firstDate);
monday.setDate(firstDate.getDate() + mondayOffset);

// Build header row for each weekday
for (let i = 0; i &amp;lt; 5; i++) {
  const dayDate = new Date(monday);
  dayDate.setDate(monday.getDate() + i);
  const dateStr = Utilities.formatDate(dayDate, Session.getScriptTimeZone(), 'MM/dd');
  const dayName = dayNames[i];

  const cell = headerRow.appendTableCell(dayName + ' (' + dateStr + ')');
  cell.setBackgroundColor('#cc0000');
  cell.getChild(0).asParagraph().setForegroundColor('#ffffff');
  cell.getChild(0).asParagraph().setAlignment(DocumentApp.HorizontalAlignment.CENTER);
  cell.setWidth(144); // 720/5 = 144 points per column
}

// Content row
const contentRow = table.appendTableRow();

// Detect conflicts (meetings at same time on same day)
for (const date of Object.keys(meetingsByDate)) {
  const dayMeetings = meetingsByDate[date];
  const timeSlots = {};

  dayMeetings.forEach(m =&amp;gt; {
    const timeKey = m.time24;
    if (!timeSlots[timeKey]) {
      timeSlots[timeKey] = [];
    }
```

CODE.GS (continued - lines 1521+)

```
      timeSlots[timeKey].push(m);
    });

    // Mark conflicts
    for (const timeKey of Object.keys(timeSlots)) {
      if (timeSlots[timeKey].length &gt; 1) {
        timeSlots[timeKey].forEach(m =&gt; {
          m.hasConflict = true;
        });
      }
    }
  }

  // Fill in content for each day
  for (let i = 0; i &lt; 5; i++) {
    const dayDate = new Date(monday);
    dayDate.setDate(monday.getDate() + i);
    const dateStr = Utilities.formatDate(dayDate, Session.getScriptTimeZone(), 'yyyy-MM-dd');

    const cell = contentRow.appendTableCell();
    cell.setVerticalAlignment(DocumentApp.VerticalAlignment.TOP);

    const dayMeetings = meetingsByDate[dateStr] || [];
    dayMeetings.sort((a, b) =&gt; (a.time24 || '').localeCompare(b.time24 || ''));

    if (dayMeetings.length === 0) {
      cell.setText('No meetings');
      cell.getChild(0).asParagraph().setForegroundColor('#999999');
    } else {
      // Clear default text
      cell.clear();

      dayMeetings.forEach((m, idx) =&gt; {
        // Time
        const timePara = cell.appendParagraph(m.time || 'TBD');
        timePara.setBold(true);
        timePara.setFontSize(9);

        // Meeting name
        const namePara = cell.appendParagraph(m.title);
        namePara.setFontSize(8);

        // Highlight important meetings in red
        if (m.isHotTopic || m.prepRequired) {
          namePara.setForegroundColor('#cc0000');
        }

        // Mark conflicts
        if (m.hasConflict) {
          const conflictPara = cell.appendParagraph('■■ CONFLICT');
          conflictPara.setForegroundColor('#ff6600');
          conflictPara.setFontSize(7);
          conflictPara.setBold(true);

          // Also highlight the meeting name
          namePara.setBackgroundColor('#fff3cd');
        }

        // Add prep meeting indicator if needed
        if (includePrepMeetings &amp;&amp; m.prepRequired) {
          const prepPara = cell.appendParagraph('■ Prep Required');
          prepPara.setFontSize(7);
          prepPara.setForegroundColor('#0066cc');
        }

        // Add spacing between meetings
        if (idx &lt; dayMeetings.length - 1) {
          cell.appendParagraph('');
        }
      });
    }
  }

  // Add agenda items section if provided
  if (agendaItems &amp;&amp; Object.keys(agendaItems).length &gt; 0) {
    body.appendParagraph('');

    // Add bottom section table
    const bottomTable = body.appendTable();
    const bottomRow = bottomTable.appendTableRow();
```

CODE.GS (continued - lines 1601+)

```
      // Add cells for each agenda section
      for (const [sectionName, items] of Object.entries(agendaItems)) {
        const cell = bottomRow.appendTableCell();
        cell.setBackgroundColor('#000000');

        // Section header
        const headerPara = cell.appendParagraph(sectionName);
        headerPara.setForegroundColor('#ffffff');
        headerPara.setBold(true);
        headerPara.setFontSize(10);

        // Items
        if (items &amp;&amp; items.length &gt; 0) {
          items.forEach(item =&gt; {
            const itemPara = cell.appendParagraph('• ' + item);
            itemPara.setForegroundColor('#ffffff');
            itemPara.setFontSize(8);
          });
        }
      }
    }

    // Add Verizon footer
    body.appendParagraph('');
    const footer = body.appendParagraph('verizon');
    footer.setFontFamily('Arial');
    footer.setBold(true);
    footer.setFontSize(14);
    footer.setForegroundColor('#cc0000');

    // Save and get PDF
    doc.saveAndClose();

    // Convert to PDF
    const docFile = DriveApp.getFileById(doc.getId());
    const pdfBlob = docFile.getAs('application/pdf');
    pdfBlob.setName('Weekly_Schedule_' + Utilities.formatDate(new Date(), Session.getScriptTimeZone(), 'yyyy-MM-dd') + '.pdf');

    // Save PDF to Drive
    const pdfFile = DriveApp.createFile(pdfBlob);

    // Delete the temporary Doc
    docFile.setTrashed(true);

    return {
      success: true,
      pdfUrl: pdfFile.getUrl(),
      pdfId: pdfFile.getId(),
      downloadUrl: 'https://drive.google.com/uc?export=download&amp;id=' + pdfFile.getId()
    };

  } catch (error) {
    Logger.log('Error generating PDF: ' + error);
    return {
      success: false,
      error: error.toString()
    };
  }
}

/**
 * Get meetings data formatted for PDF export selection
 */
function getMeetingsForPDFExport() {
  try {
    const meetings = getMeetings();

    // Group by date and detect conflicts
    const meetingsByDate = {};
    meetings.forEach(m =&gt; {
      if (!meetingsByDate[m.date]) {
        meetingsByDate[m.date] = [];
      }
      meetingsByDate[m.date].push(m);
    });

    // Detect conflicts
    for (const date of Object.keys(meetingsByDate)) {
      const dayMeetings = meetingsByDate[date];
```

CODE.GS (continued - lines 1681+)

```
      const timeSlots = {};

      dayMeetings.forEach(m =&gt; {
        const timeKey = m.time24;
        if (!timeSlots[timeKey]) {
          timeSlots[timeKey] = [];
        }
        timeSlots[timeKey].push(m);
      });

      // Mark conflicts
      for (const timeKey of Object.keys(timeSlots)) {
        if (timeSlots[timeKey].length &gt; 1) {
          timeSlots[timeKey].forEach(m =&gt; {
            m.hasConflict = true;
            m.conflictCount = timeSlots[timeKey].length;
          });
        }
      }
    }

    return {
      success: true,
      meetings: meetings,
      meetingsByDate: meetingsByDate
    };

  } catch (error) {
    Logger.log('Error getting meetings for PDF: ' + error);
    return {
      success: false,
      error: error.toString()
    };
  }
}

// ============================================================================
// HELPER FUNCTIONS
// ============================================================================

/**
 * Get sheet by ID and tab name
 */
function getSheet(sheetId, tabName) {
  const ss = SpreadsheetApp.openById(sheetId);
  return ss.getSheetByName(tabName);
}

/**
 * Calculate urgency level
 */
function getUrgencyLevel(daysRemaining) {
  if (daysRemaining === null) return 'none';
  if (daysRemaining &lt; 0) return 'overdue';
  if (daysRemaining &lt;= 5) return 'critical';
  if (daysRemaining &lt;= 10) return 'high';
  if (daysRemaining &lt;= 20) return 'medium';
  return 'low';
}

/**
 * Test function
 */
function testDataRetrieval() {
  Logger.log('Testing data retrieval...');
  Logger.log('Exec 1 Deliverables:', JSON.stringify(getExec1Deliverables(), null, 2));
  Logger.log('Meetings:', JSON.stringify(getMeetings(), null, 2));
  Logger.log('Recurring Meetings:', JSON.stringify(getRecurringMeetings(), null, 2));
}

// ============================================================================
// V8 NEW FUNCTIONS
// ============================================================================

/**
 * Get Document Repository Links (separate from main document list)
 */
function getDocumentRepoLinks() {
  try {
    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);
```

CODE.GS (continued - lines 1761+)

```
    const startRow = CONFIG.DOCUMENTS_START_ROW;
    const maxRows = CONFIG.DOCUMENTS_END_ROW - startRow + 1;

    const range = sheet.getRange(startRow, 1, maxRows, 7);
    const values = range.getValues();
    const richTextValues = range.getRichTextValues();

    const links = [];
    for (let i = 0; i &lt; values.length; i++) {
      const row = values[i];
      if (!row[0] || row[0].toString().trim() === '') break;

      // Extract URL from Column D (index 3) hyperlink
      let url = '';
      const richText = richTextValues[i][3];
      if (richText) {
        url = richText.getLinkUrl() || '';
      }
      if (!url &amp;&amp; row[3]) {
        url = String(row[3]);
      }

      // Column mapping: A=Description (0), C=Owner (2), D=Link (3), G=Comment (6)
      links.push({
        id: 'doc-repo-' + i,
        name: String(row[0] || ''),           // Column A - Description
        description: String(row[0] || ''),    // Column A - Description
        url: url,                              // Column D - Link to Materials
        link: url,                             // Column D - Link to Materials
        owner: String(row[2] || ''),          // Column C - Owner
        comment: String(row[6] || ''),        // Column G - Comment (was row[4], now row[6])
        category: 'document-repo'
      });
    }

    return links;
  } catch (error) {
    Logger.log('Error getting document repo links: ' + error);
    return [];
  }
}

/**
 * Complete an item and move it to destination
 */
function completeItem(itemId, destination) {
  try {
    Logger.log('Completing item: ' + itemId + ' to ' + destination);

    // This is a placeholder - implement based on your sheet structure
    // You'll need to:
    // 1. Find the item by ID in the appropriate sheet section
    // 2. Copy it to the destination section
    // 3. Delete or mark it as complete in the source

    return { success: true, message: 'Item completed and moved to ' + destination };

  } catch (error) {
    Logger.log('Error completing item: ' + error);
    throw error;
  }
}

/**
 * Add new item to deliverables
 */
function addNewItem(itemData) {
  try {
    Logger.log('Adding new item to: ' + itemData.section);

    const sheet = getSheet(CONFIG.DELIVERABLES_SHEET_ID, CONFIG.DELIVERABLES_TAB_NAME);

    // Determine target section
    let targetStartRow;
    if (itemData.section === 'exec1') {
      targetStartRow = CONFIG.EXEC1_START_ROW;
    } else if (itemData.section === 'exec2') {
      targetStartRow = CONFIG.EXEC2_START_ROW;
    } else {
      throw new Error('Unknown section: ' + itemData.section);
```

CODE.GS (continued - lines 1841+)

```
    }

    // Find first empty row in section
    let targetRow = targetStartRow;
    while (sheet.getRange(targetRow, 1).getValue() !== '') {
      targetRow++;
    }

    // Prepare row data (adjust columns based on your sheet structure)
    const rowData = [
      itemData.description,
      itemData.eta || '',
      itemData.owner || '',
      itemData.link || '',
      itemData.comments || '',
      itemData.isHotTopic ? 'YES' : ''
    ];

    // Insert data
    sheet.getRange(targetRow, 1, 1, rowData.length).setValues([rowData]);

    return { success: true, message: 'Item added successfully' };

  } catch (error) {
    Logger.log('Error adding item: ' + error);
    throw error;
  }
}

// ============================================================================
// DELIVERABLE CRUD FUNCTIONS
// ============================================================================

/**
 * Update a deliverable item
 * @param {Object} data - Deliverable data including execType, itemIndex, and all fields
 * @return {Object} Success/error response
 */
function updateDeliverable(data) {
  try {
    const ss = SpreadsheetApp.openById(CONFIG.DELIVERABLES_SHEET_ID);
    const sheet = ss.getSheetByName(CONFIG.DELIVERABLES_TAB_NAME);

    if (!sheet) {
      return { success: false, error: 'Sheet not found: ' + CONFIG.DELIVERABLES_TAB_NAME };
    }

    // Determine the row range based on execType
    let startRow, endRow;
    if (data.execType === 'exec1') {
      startRow = CONFIG.EXEC1_START_ROW;
      endRow = CONFIG.EXEC1_END_ROW;
    } else if (data.execType === 'exec2') {
      startRow = CONFIG.EXEC2_START_ROW;
      endRow = CONFIG.EXEC2_END_ROW;
    } else {
      return { success: false, error: 'Invalid execType: ' + data.execType };
    }

    // Calculate actual row number
    const actualRow = startRow + data.itemIndex;

    if (actualRow &gt; endRow) {
      return { success: false, error: 'Item index out of range' };
    }

    // Update the row
    // Assuming columns: A=Description, B=ETA, C=Owner, D=Link, E=HotTopic, F=Comments, G=LinkedMeetingId, H=MeetingType, I=DueDate
    const range = sheet.getRange(actualRow, 1, 1, 9);
    const values = [[
      data.description || '',
      data.eta || '',
      data.owner || '',
      data.link || '',
      data.hotTopic ? 'TRUE' : 'FALSE',
      data.comments || '',
      data.linkedMeetingId || '',
      data.meetingType || '',
      data.dueDate || ''
    ]];
```

CODE.GS (continued - lines 1921+)

```
    range.setValues(values);

    // Update hyperlink in Column D if provided
    if (data.link) {
      const linkCell = sheet.getRange(actualRow, 4); // Column D
      linkCell.setFormula('=HYPERLINK("' + data.link + '", "Link to File")');
    }

    Logger.log('■ Updated deliverable: ' + data.description);
    return { success: true, message: 'Deliverable updated successfully' };

  } catch (error) {
    Logger.log('■ Error updating deliverable: ' + error.message);
    return { success: false, error: error.message };
  }
}

/**
 * Delete a deliverable item
 * @param {Object} data - Contains execType and itemIndex
 * @return {Object} Success/error response
 */
function deleteDeliverable(data) {
  try {
    const ss = SpreadsheetApp.openById(CONFIG.DELIVERABLES_SHEET_ID);
    const sheet = ss.getSheetByName(CONFIG.DELIVERABLES_TAB_NAME);

    if (!sheet) {
      return { success: false, error: 'Sheet not found: ' + CONFIG.DELIVERABLES_TAB_NAME };
    }

    // Determine the row range based on execType
    let startRow, endRow;
    if (data.execType === 'exec1') {
      startRow = CONFIG.EXEC1_START_ROW;
      endRow = CONFIG.EXEC1_END_ROW;
    } else if (data.execType === 'exec2') {
      startRow = CONFIG.EXEC2_START_ROW;
      endRow = CONFIG.EXEC2_END_ROW;
    } else {
      return { success: false, error: 'Invalid execType: ' + data.execType };
    }

    // Calculate actual row number
    const actualRow = startRow + data.itemIndex;

    if (actualRow &gt; endRow) {
      return { success: false, error: 'Item index out of range' };
    }

    // Clear the row (don't delete to preserve row numbers)
    const range = sheet.getRange(actualRow, 1, 1, 9);
    range.clearContent();

    Logger.log('■ Deleted deliverable at row: ' + actualRow);
    return { success: true, message: 'Deliverable deleted successfully' };

  } catch (error) {
    Logger.log('■ Error deleting deliverable: ' + error.message);
    return { success: false, error: error.message };
  }
}

/**
 * Add a new deliverable item
 * @param {Object} data - Deliverable data including execType and all fields
 * @return {Object} Success/error response
 */
function addDeliverable(data) {
  try {
    const ss = SpreadsheetApp.openById(CONFIG.DELIVERABLES_SHEET_ID);
    const sheet = ss.getSheetByName(CONFIG.DELIVERABLES_TAB_NAME);

    if (!sheet) {
      return { success: false, error: 'Sheet not found: ' + CONFIG.DELIVERABLES_TAB_NAME };
    }

    // Determine the row range based on execType
    let startRow, endRow;
```

CODE.GS (continued - lines 2001+)

```
    if (data.execType === 'exec1') {
      startRow = CONFIG.EXEC1_START_ROW;
      endRow = CONFIG.EXEC1_END_ROW;
    } else if (data.execType === 'exec2') {
      startRow = CONFIG.EXEC2_START_ROW;
      endRow = CONFIG.EXEC2_END_ROW;
    } else {
      return { success: false, error: 'Invalid execType: ' + data.execType };
    }

    // Find the first empty row in the range
    const range = sheet.getRange(startRow, 1, endRow - startRow + 1, 1);
    const values = range.getValues();
    let emptyRow = null;

    for (let i = 0; i &lt; values.length; i++) {
      if (!values[i][0] || values[i][0].toString().trim() === '') {
        emptyRow = startRow + i;
        break;
      }
    }

    if (!emptyRow) {
      return { success: false, error: 'No empty rows available in ' + data.execType + ' section' };
    }

    // Add the new item
    const newRange = sheet.getRange(emptyRow, 1, 1, 9);
    const newValues = [[
      data.description || '',
      data.eta || '',
      data.owner || '',
      data.link || '',
      data.hotTopic ? 'TRUE' : 'FALSE',
      data.comments || '',
      data.linkedMeetingId || '',
      data.meetingType || '',
      data.dueDate || ''
    ]];

    newRange.setValues(newValues);

    // Set hyperlink in Column D if provided
    if (data.link) {
      const linkCell = sheet.getRange(emptyRow, 4); // Column D
      linkCell.setFormula('=HYPERLINK("' + data.link + '", "Link to File")');
    }

    Logger.log('■ Added new deliverable: ' + data.description + ' at row ' + emptyRow);
    return { success: true, message: 'Deliverable added successfully', row: emptyRow };

  } catch (error) {
    Logger.log('■ Error adding deliverable: ' + error.message);
    return { success: false, error: error.message };
  }
}

/**
 * Update a meeting
 */
function updateMeeting(meetingId, data) {
  try {
    const sheet = getSheet(CONFIG.MEETINGS_SHEET_ID, CONFIG.MEETINGS_TAB_NAME);
    if (!sheet) {
      return { success: false, error: 'Meetings sheet not found' };
    }

    const dataRange = sheet.getDataRange();
    const values = dataRange.getValues();

    // Find the meeting row
    let meetingRow = -1;
    for (let i = 1; i &lt; values.length; i++) {
      if (values[i][CONFIG.MEETING_COLUMNS.EVENT_ID] === meetingId) {
        meetingRow = i + 1; // 1-indexed
        break;
      }
    }

    if (meetingRow === -1) {
```

CODE.GS (continued - lines 2081+)

```
      return { success: false, error: 'Meeting not found: ' + meetingId };
    }

    // Update the fields
    if (data.notes !== undefined) {
      sheet.getRange(meetingRow, CONFIG.MEETING_COLUMNS.NOTES + 1).setValue(data.notes);
    }
    if (data.category !== undefined) {
      sheet.getRange(meetingRow, CONFIG.MEETING_COLUMNS.CATEGORY + 1).setValue(data.category);
    }
    if (data.deliverableLink !== undefined) {
      sheet.getRange(meetingRow, CONFIG.MEETING_COLUMNS.DELIVERABLE_LINK + 1).setValue(data.deliverableLink);
    }

    Logger.log('■ Updated meeting: ' + meetingId);
    return { success: true, message: 'Meeting updated successfully' };

  } catch (error) {
    Logger.log('■ Error updating meeting: ' + error);
    return { success: false, error: error.toString() };
  }
}

/**
 * Delete a meeting
 */
function deleteMeeting(meetingId) {
  try {
    const sheet = getSheet(CONFIG.MEETINGS_SHEET_ID, CONFIG.MEETINGS_TAB_NAME);
    if (!sheet) {
      return { success: false, error: 'Meetings sheet not found' };
    }

    const dataRange = sheet.getDataRange();
    const values = dataRange.getValues();

    // Find the meeting row
    let meetingRow = -1;
    for (let i = 1; i &lt; values.length; i++) {
      if (values[i][CONFIG.MEETING_COLUMNS.EVENT_ID] === meetingId) {
        meetingRow = i + 1; // 1-indexed
        break;
      }
    }

    if (meetingRow === -1) {
      return { success: false, error: 'Meeting not found: ' + meetingId };
    }

    // Delete the row
    sheet.deleteRow(meetingRow);

    Logger.log('■ Deleted meeting: ' + meetingId);
    return { success: true, message: 'Meeting deleted successfully' };

  } catch (error) {
    Logger.log('■ Error deleting meeting: ' + error);
    return { success: false, error: error.toString() };
  }
}
```

5. INDEX.HTML (FRONTEND)

Dashboard UI - 1,563 lines

Location: /mnt/user-data/outputs/index.html

```
&lt;!DOCTYPE html&gt;
&lt;html&gt;
&lt;head&gt;
  &lt;base target="_top"&gt;
  &lt;meta charset="UTF-8"&gt;
  &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
  &lt;title&gt;Value Governance Dashboard&lt;/title&gt;
  &lt;link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&amp;display=swap" rel="stylesheet"&gt;
  &lt;link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" rel="stylesheet"&gt;
  &lt;style&gt;
    :root {
      --verizon-red: #cd040b;
      --verizon-red-dark: #a00308;
      --verizon-black: #000000;
      --verizon-white: #ffffff;
      --gray-50: #f9fafb;
      --gray-100: #f3f4f6;
      --gray-200: #e5e7eb;
      --gray-300: #d1d5db;
      --gray-400: #9ca3af;
      --gray-500: #6b7280;
      --gray-600: #4b5563;
      --gray-700: #374151;
      --gray-800: #1f2937;
      --gray-900: #111827;
      --success: #10b981;
      --warning: #f59e0b;
      --danger: #ef4444;
      --info: #3b82f6;
    }

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

    body {
      font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
      background: var(--gray-100);
      color: var(--gray-900);
      min-height: 100vh;
    }

    /* Main Layout */
    .dashboard-container {
      display: flex;
      min-height: 100vh;
    }

    /* Left Sidebar */
    .sidebar {
      width: 280px;
      background: var(--verizon-black);
      color: var(--verizon-white);
      display: flex;
      flex-direction: column;
      position: fixed;
      height: 100vh;
      overflow-y: auto;
      z-index: 100;
    }

    .sidebar-header {
      padding: 20px;
      border-bottom: 1px solid rgba(255,255,255,0.1);
    }

    .logo {
      font-size: 28px;
      font-weight: 700;
      color: var(--verizon-red);
      letter-spacing: -1px;
    }

    .logo-subtitle {
      font-size: 11px;
      color: var(--gray-400);
      margin-top: 4px;
      text-transform: uppercase;
```

INDEX.HTML (continued - lines 81+)

```
  letter-spacing: 1px;
}

.sidebar-nav {
  padding: 20px 0;
  flex: 1;
}

.nav-item {
  display: flex;
  align-items: center;
  padding: 12px 20px;
  color: var(--gray-300);
  cursor: pointer;
  transition: all 0.2s;
  border-left: 3px solid transparent;
}

.nav-item:hover {
  background: rgba(255,255,255,0.05);
  color: var(--verizon-white);
}

.nav-item.active {
  background: rgba(205,4,11,0.1);
  color: var(--verizon-red);
  border-left-color: var(--verizon-red);
}

.nav-item i {
  width: 20px;
  margin-right: 12px;
}

.nav-section-title {
  padding: 20px 20px 10px;
  font-size: 11px;
  text-transform: uppercase;
  letter-spacing: 1px;
  color: var(--gray-500);
}

/* Deliverables in Sidebar */
.sidebar-deliverables {
  padding: 0 12px 20px;
}

.sidebar-deliverable-item {
  padding: 10px 12px;
  border-radius: 6px;
  margin-bottom: 4px;
  cursor: pointer;
  transition: all 0.2s;
  border-left: 3px solid transparent;
}

.sidebar-deliverable-item:hover {
  background: rgba(255,255,255,0.05);
}

.sidebar-deliverable-item.critical {
  border-left-color: var(--danger);
}

.sidebar-deliverable-item.high {
  border-left-color: var(--warning);
}

.sidebar-deliverable-item.medium {
  border-left-color: var(--info);
}

.sidebar-deliverable-item .item-title {
  font-size: 13px;
  color: var(--gray-200);
  margin-bottom: 4px;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
  overflow: hidden;
```

INDEX.HTML (continued - lines 161+)

```
}

.sidebar-deliverable-item .item-meta {
  font-size: 11px;
  color: var(--gray-500);
}

.sidebar-deliverable-item .item-meta .days-badge {
  display: inline-block;
  padding: 2px 6px;
  border-radius: 4px;
  font-weight: 600;
  margin-left: 6px;
}

.sidebar-deliverable-item .item-meta .days-badge.critical {
  background: rgba(239,68,68,0.2);
  color: #fca5a5;
}

.sidebar-deliverable-item .item-meta .days-badge.high {
  background: rgba(245,158,11,0.2);
  color: #fcd34d;
}

/* Main Content */
.main-content {
  flex: 1;
  margin-left: 280px;
  display: flex;
  flex-direction: column;
}

/* Top Header */
.top-header {
  background: var(--verizon-white);
  padding: 16px 24px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  border-bottom: 1px solid var(--gray-200);
  position: sticky;
  top: 0;
  z-index: 50;
}

.page-title {
  font-size: 20px;
  font-weight: 600;
}

.header-actions {
  display: flex;
  gap: 12px;
  align-items: center;
}

.btn {
  display: inline-flex;
  align-items: center;
  gap: 8px;
  padding: 8px 16px;
  border-radius: 6px;
  font-size: 14px;
  font-weight: 500;
  cursor: pointer;
  border: none;
  transition: all 0.2s;
}

.btn-primary {
  background: var(--verizon-red);
  color: var(--verizon-white);
}

.btn-primary:hover {
  background: var(--verizon-red-dark);
}

.btn-secondary {
```

INDEX.HTML (continued - lines 241+)

```
      background: var(--gray-100);
      color: var(--gray-700);
      border: 1px solid var(--gray-300);
    }

    .btn-secondary:hover {
      background: var(--gray-200);
    }

    /* Content Area */
    .content-area {
      padding: 24px;
      flex: 1;
    }

    /* Cards */
    .card {
      background: var(--verizon-white);
      border-radius: 12px;
      box-shadow: 0 1px 3px rgba(0,0,0,0.1);
      overflow: hidden;
    }

    .card-header {
      padding: 16px 20px;
      border-bottom: 1px solid var(--gray-200);
      display: flex;
      justify-content: space-between;
      align-items: center;
    }

    .card-title {
      font-size: 16px;
      font-weight: 600;
    }

    .card-body {
      padding: 20px;
    }

    /* Grid Layout */
    .grid {
      display: grid;
      gap: 24px;
    }

    .grid-2 {
      grid-template-columns: repeat(2, 1fr);
    }

    .grid-3 {
      grid-template-columns: repeat(3, 1fr);
    }

    /* Stats Cards */
    .stat-card {
      padding: 20px;
      background: var(--verizon-white);
      border-radius: 12px;
      box-shadow: 0 1px 3px rgba(0,0,0,0.1);
    }

    .stat-value {
      font-size: 32px;
      font-weight: 700;
      color: var(--gray-900);
    }

    .stat-label {
      font-size: 13px;
      color: var(--gray-500);
      margin-top: 4px;
    }

    .stat-trend {
      display: inline-flex;
      align-items: center;
      gap: 4px;
      font-size: 12px;
      margin-top: 8px;
```

INDEX.HTML (continued - lines 321+)

```
      padding: 4px 8px;
      border-radius: 4px;
    }

    .stat-trend.up {
      background: rgba(16,185,129,0.1);
      color: var(--success);
    }

    .stat-trend.down {
      background: rgba(239,68,68,0.1);
      color: var(--danger);
    }

    /* Table Styles */
    .table-container {
      overflow-x: auto;
    }

    table {
      width: 100%;
      border-collapse: collapse;
    }

    th, td {
      padding: 12px 16px;
      text-align: left;
      border-bottom: 1px solid var(--gray-200);
    }

    th {
      font-size: 12px;
      font-weight: 600;
      text-transform: uppercase;
      letter-spacing: 0.5px;
      color: var(--gray-500);
      background: var(--gray-50);
    }

    td {
      font-size: 14px;
    }

    tr:hover {
      background: var(--gray-50);
    }

    /* Status Badges */
    .badge {
      display: inline-flex;
      align-items: center;
      padding: 4px 10px;
      border-radius: 20px;
      font-size: 12px;
      font-weight: 500;
    }

    .badge-critical {
      background: rgba(239,68,68,0.1);
      color: var(--danger);
    }

    .badge-high {
      background: rgba(245,158,11,0.1);
      color: var(--warning);
    }

    .badge-medium {
      background: rgba(59,130,246,0.1);
      color: var(--info);
    }

    .badge-low {
      background: rgba(16,185,129,0.1);
      color: var(--success);
    }

    .badge-text {
      background: rgba(107,114,128,0.1);
      color: var(--gray-600);
```

INDEX.HTML (continued - lines 401+)

```
}

/* Loading State */
.loading {
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 60px;
  color: var(--gray-500);
}

.loading i {
  animation: spin 1s linear infinite;
  margin-right: 8px;
}

@keyframes spin {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

/* Empty State */
.empty-state {
  text-align: center;
  padding: 40px;
  color: var(--gray-500);
}

.empty-state i {
  font-size: 48px;
  margin-bottom: 16px;
  color: var(--gray-300);
}

/* Schedule Grid */
.schedule-grid {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
  gap: 16px;
}

.schedule-day {
  background: var(--gray-50);
  border-radius: 8px;
  overflow: hidden;
}

.schedule-day-header {
  background: var(--verizon-black);
  color: var(--verizon-white);
  padding: 12px;
  text-align: center;
}

.schedule-day-header .day-name {
  font-weight: 600;
  font-size: 14px;
}

.schedule-day-header .day-date {
  font-size: 12px;
  color: var(--gray-400);
  margin-top: 2px;
}

.schedule-day-content {
  padding: 12px;
  min-height: 200px;
}

.schedule-meeting {
  background: var(--verizon-white);
  border-radius: 6px;
  padding: 10px;
  margin-bottom: 8px;
  border-left: 3px solid var(--gray-300);
  cursor: pointer;
  transition: all 0.2s;
}
```

INDEX.HTML (continued - lines 481+)

```
.schedule-meeting:hover {
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
}

.schedule-meeting.hot-topic {
  border-left-color: var(--verizon-red);
}

.schedule-meeting.prep-required {
  border-left-color: var(--warning);
}

.schedule-meeting.has-conflict {
  background: #fff3cd;
  border-left-color: #ff6600;
}

.meeting-time {
  font-size: 11px;
  font-weight: 600;
  color: var(--gray-500);
}

.meeting-title {
  font-size: 13px;
  font-weight: 500;
  margin-top: 4px;
  color: var(--gray-800);
}

.meeting-badges {
  display: flex;
  gap: 4px;
  margin-top: 6px;
  flex-wrap: wrap;
}

.meeting-badge {
  font-size: 9px;
  padding: 2px 6px;
  border-radius: 4px;
  font-weight: 600;
}

.meeting-badge.hot {
  background: var(--verizon-red);
  color: white;
}

.meeting-badge.prep {
  background: var(--warning);
  color: white;
}

.meeting-badge.conflict {
  background: #ff6600;
  color: white;
}

/* Week Tabs */
.week-tabs {
  display: flex;
  gap: 8px;
  margin-bottom: 16px;
}

.week-tab {
  padding: 8px 16px;
  border-radius: 6px;
  font-size: 14px;
  font-weight: 500;
  cursor: pointer;
  background: var(--gray-100);
  color: var(--gray-600);
  border: none;
  transition: all 0.2s;
}

.week-tab.active {
  background: var(--verizon-red);
```

INDEX.HTML (continued - lines 561+)

```
  color: white;
}

/* Links Grid */
.links-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  gap: 12px;
}

.link-card {
  display: flex;
  align-items: center;
  padding: 12px 16px;
  background: var(--gray-50);
  border-radius: 8px;
  text-decoration: none;
  color: var(--gray-700);
  transition: all 0.2s;
  border: 1px solid transparent;
}

.link-card:hover {
  background: var(--verizon-white);
  border-color: var(--gray-200);
  box-shadow: 0 2px 8px rgba(0,0,0,0.05);
}

.link-card i {
  width: 32px;
  height: 32px;
  display: flex;
  align-items: center;
  justify-content: center;
  background: var(--verizon-red);
  color: white;
  border-radius: 6px;
  margin-right: 12px;
}

.link-card span {
  font-size: 14px;
  font-weight: 500;
}

/* Tabs */
.tabs {
  display: flex;
  border-bottom: 1px solid var(--gray-200);
  margin-bottom: 20px;
}

.tab {
  padding: 12px 20px;
  font-size: 14px;
  font-weight: 500;
  color: var(--gray-500);
  cursor: pointer;
  border-bottom: 2px solid transparent;
  transition: all 0.2s;
}

.tab:hover {
  color: var(--gray-700);
}

.tab.active {
  color: var(--verizon-red);
  border-bottom-color: var(--verizon-red);
}

/* Document Repository Section */
.doc-repo-section {
  margin-top: 24px;
}

.doc-repo-list {
  display: grid;
  gap: 8px;
}
```

INDEX.HTML (continued - lines 641+)

```
.doc-repo-item {
  display: flex;
  align-items: center;
  padding: 12px 16px;
  background: var(--gray-50);
  border-radius: 8px;
  transition: all 0.2s;
}

.doc-repo-item:hover {
  background: var(--verizon-white);
  box-shadow: 0 2px 4px rgba(0,0,0,0.05);
}

.doc-repo-item i {
  color: var(--verizon-red);
  margin-right: 12px;
}

.doc-repo-item .doc-info {
  flex: 1;
}

.doc-repo-item .doc-name {
  font-weight: 500;
  color: var(--gray-800);
}

.doc-repo-item .doc-meta {
  font-size: 12px;
  color: var(--gray-500);
  margin-top: 2px;
}

.doc-repo-item .doc-link {
  color: var(--verizon-red);
  text-decoration: none;
}

.doc-repo-item .doc-link:hover {
  text-decoration: underline;
}

/* PDF Export Modal */
.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0,0,0,0.5);
  z-index: 1000;
  align-items: center;
  justify-content: center;
}

.modal.active {
  display: flex;
}

.modal-content {
  background: white;
  border-radius: 12px;
  width: 90%;
  max-width: 800px;
  max-height: 90vh;
  overflow-y: auto;
}

.modal-header {
  padding: 20px;
  border-bottom: 1px solid var(--gray-200);
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.modal-header h2 {
```

INDEX.HTML (continued - lines 721+)

```
      font-size: 18px;
      font-weight: 600;
    }

    .modal-close {
      background: none;
      border: none;
      font-size: 20px;
      cursor: pointer;
      color: var(--gray-500);
    }

    .modal-body {
      padding: 20px;
    }

    .modal-footer {
      padding: 16px 20px;
      border-top: 1px solid var(--gray-200);
      display: flex;
      justify-content: flex-end;
      gap: 12px;
    }

    /* Alert Banner */
    .alert-banner {
      background: var(--verizon-red);
      color: white;
      padding: 10px 20px;
      font-size: 14px;
      overflow: hidden;
      display: none;
    }

    .alert-banner.active {
      display: block;
    }

    .alert-banner .alert-text {
      animation: scroll-left 20s linear infinite;
      white-space: nowrap;
    }

    @keyframes scroll-left {
      0% { transform: translateX(100%); }
      100% { transform: translateX(-100%); }
    }

    /* Responsive */
    @media (max-width: 1024px) {
      .sidebar {
        width: 60px;
      }
      .sidebar-header, .nav-section-title, .sidebar-deliverables {
        display: none;
      }
      .nav-item span {
        display: none;
      }
      .nav-item {
        justify-content: center;
        padding: 16px;
      }
      .nav-item i {
        margin-right: 0;
      }
      .main-content {
        margin-left: 60px;
      }
      .schedule-grid {
        grid-template-columns: 1fr;
      }
    }
  &lt;/style&gt;
&lt;/head&gt;
&lt;body&gt;
  &lt;div class="dashboard-container"&gt;
    &lt;!-- Sidebar --&gt;
    &lt;aside class="sidebar"&gt;
      &lt;div class="sidebar-header"&gt;
```

INDEX.HTML (continued - lines 801+)

```
        &lt;div class="logo"&gt;verizon&lt;/div&gt;
        &lt;div class="logo-subtitle"&gt;Value Governance&lt;/div&gt;
      &lt;/div&gt;

      &lt;nav class="sidebar-nav"&gt;
        &lt;div class="nav-item active" data-page="overview"&gt;
          &lt;i class="fas fa-home"&gt;&lt;/i&gt;
          &lt;span&gt;Overview&lt;/span&gt;
        &lt;/div&gt;
        &lt;div class="nav-item" data-page="schedule"&gt;
          &lt;i class="fas fa-calendar-alt"&gt;&lt;/i&gt;
          &lt;span&gt;Schedule&lt;/span&gt;
        &lt;/div&gt;
        &lt;div class="nav-item" data-page="deliverables"&gt;
          &lt;i class="fas fa-tasks"&gt;&lt;/i&gt;
          &lt;span&gt;Deliverables&lt;/span&gt;
        &lt;/div&gt;
        &lt;div class="nav-item" data-page="documents"&gt;
          &lt;i class="fas fa-folder-open"&gt;&lt;/i&gt;
          &lt;span&gt;Documents&lt;/span&gt;
        &lt;/div&gt;
        &lt;div class="nav-item" data-page="links"&gt;
          &lt;i class="fas fa-link"&gt;&lt;/i&gt;
          &lt;span&gt;Quick Links&lt;/span&gt;
        &lt;/div&gt;

        &lt;div class="nav-section-title"&gt;Upcoming Deadlines&lt;/div&gt;
        &lt;div class="sidebar-deliverables" id="sidebar-deliverables"&gt;
          &lt;!-- Populated by JS --&gt;
        &lt;/div&gt;
      &lt;/nav&gt;
    &lt;/aside&gt;

    &lt;!-- Main Content --&gt;
    &lt;main class="main-content"&gt;
      &lt;!-- Alert Banner --&gt;
      &lt;div class="alert-banner" id="alert-banner"&gt;
        &lt;div class="alert-text" id="alert-text"&gt;&lt;/div&gt;
      &lt;/div&gt;

      &lt;!-- Top Header --&gt;
      &lt;header class="top-header"&gt;
        &lt;h1 class="page-title" id="page-title"&gt;Overview&lt;/h1&gt;
        &lt;div class="header-actions"&gt;
          &lt;button class="btn btn-secondary" onclick="refreshData()"&gt;
            &lt;i class="fas fa-sync-alt"&gt;&lt;/i&gt; Refresh
          &lt;/button&gt;
          &lt;button class="btn btn-primary" onclick="openPDFExportModal()"&gt;
            &lt;i class="fas fa-file-pdf"&gt;&lt;/i&gt; Export PDF
          &lt;/button&gt;
        &lt;/div&gt;
      &lt;/header&gt;

      &lt;!-- Content Area --&gt;
      &lt;div class="content-area" id="content-area"&gt;
        &lt;!-- Content loaded dynamically --&gt;
      &lt;/div&gt;
    &lt;/main&gt;
  &lt;/div&gt;

  &lt;!-- PDF Export Modal --&gt;
  &lt;div class="modal" id="pdf-modal"&gt;
    &lt;div class="modal-content"&gt;
      &lt;div class="modal-header"&gt;
        &lt;h2&gt;Export Weekly Schedule PDF&lt;/h2&gt;
        &lt;button class="modal-close" onclick="closePDFModal()"&gt;&amp;times;&lt;/button&gt;
      &lt;/div&gt;
      &lt;div class="modal-body" id="pdf-modal-body"&gt;
        &lt;!-- Populated by JS --&gt;
      &lt;/div&gt;
      &lt;div class="modal-footer"&gt;
        &lt;button class="btn btn-secondary" onclick="closePDFModal()"&gt;Cancel&lt;/button&gt;
        &lt;button class="btn btn-primary" onclick="generatePDF()"&gt;
          &lt;i class="fas fa-download"&gt;&lt;/i&gt; Generate PDF
        &lt;/button&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  &lt;/div&gt;

  &lt;script&gt;
```

INDEX.HTML (continued - lines 881+)

```
// Global data store
let dashboardData = {
  exec1Deliverables: [],
  exec2Deliverables: [],
  meetings: [],
  recurringMeetings: [],
  adminLinks: [],
  kpiLinks: [],
  executiveUpdates: [],
  documentRepository: [],
  documentRepoLinks: [],
  completedItems: [],
  alerts: []
};

let currentPage = 'overview';
let currentWeek = 1;

// Initialize
document.addEventListener('DOMContentLoaded', () =&gt; {
  loadDashboardData();
  setupNavigation();
});

// Setup navigation
function setupNavigation() {
  document.querySelectorAll('.nav-item').forEach(item =&gt; {
    item.addEventListener('click', () =&gt; {
      const page = item.dataset.page;
      if (page) {
        navigateTo(page);
      }
    });
  });
}

// Navigate to page
function navigateTo(page) {
  currentPage = page;

  // Update nav
  document.querySelectorAll('.nav-item').forEach(item =&gt; {
    item.classList.toggle('active', item.dataset.page === page);
  });

  // Update title
  const titles = {
    overview: 'Overview',
    schedule: 'Meeting Schedule',
    deliverables: 'Deliverables',
    documents: 'Document Repository',
    links: 'Quick Links'
  };
  document.getElementById('page-title').textContent = titles[page] || 'Dashboard';

  // Render page
  renderPage(page);
}

// Load dashboard data
function loadDashboardData() {
  showLoading();

  google.script.run
    .withSuccessHandler(data =&gt; {
      dashboardData = data;
      renderSidebarDeliverables();
      renderPage(currentPage);
      loadAlerts();
    })
    .withFailureHandler(error =&gt; {
      console.error('Error loading data:', error);
      showError('Failed to load dashboard data');
    })
    .getAllDashboardData();
}

// Refresh data
function refreshData() {
  loadDashboardData();
```

INDEX.HTML (continued - lines 961+)

```
}

// Load alerts
function loadAlerts() {
  google.script.run
    .withSuccessHandler(alerts =&gt; {
      if (alerts &amp;&amp; alerts.length &gt; 0) {
        document.getElementById('alert-text').textContent = alerts.join(' • ');
        document.getElementById('alert-banner').classList.add('active');
      }
    })
    .getAlerts();
}

// Show loading
function showLoading() {
  document.getElementById('content-area').innerHTML = `
    &lt;div class="loading"&gt;
      &lt;i class="fas fa-spinner"&gt;&lt;/i&gt; Loading...
    &lt;/div&gt;
  `;
}

// Show error
function showError(message) {
  document.getElementById('content-area').innerHTML = `
    &lt;div class="empty-state"&gt;
      &lt;i class="fas fa-exclamation-triangle"&gt;&lt;/i&gt;
      &lt;p&gt;${message}&lt;/p&gt;
    &lt;/div&gt;
  `;
}

// Render sidebar deliverables
function renderSidebarDeliverables() {
  const container = document.getElementById('sidebar-deliverables');
  const allDeliverables = [...dashboardData.exec1Deliverables, ...dashboardData.exec2Deliverables];

  // Sort by urgency and filter to upcoming
  const upcoming = allDeliverables
    .filter(d =&gt; d.daysRemaining !== null &amp;&amp; d.daysRemaining &lt;= 14)
    .sort((a, b) =&gt; (a.daysRemaining || 999) - (b.daysRemaining || 999))
    .slice(0, 5);

  if (upcoming.length === 0) {
    container.innerHTML = '&lt;div style="padding: 12px; color: var(--gray-500); font-size: 13px;"&gt;No upcoming deadlines&lt;/div&gt;';
    return;
  }

  container.innerHTML = upcoming.map(d =&gt; {
    // FIX: Only show date if ETA exists and daysRemaining is numeric
    const hasValidEta = d.eta &amp;&amp; d.daysRemaining !== null &amp;&amp; typeof d.daysRemaining === 'number';
    const etaDisplay = hasValidEta ? `&lt;span class="item-meta"&gt;${d.eta}&lt;/span&gt;` : '';
    const daysBadge = hasValidEta ?
      `&lt;span class="days-badge ${d.urgency}"&gt;${Math.abs(d.daysRemaining)}d ${d.daysRemaining &lt; 0 ? 'overdue' : 'left'}&lt;/span&gt;` : '';

    return `
      &lt;div class="sidebar-deliverable-item ${d.urgency}" onclick="navigateTo('deliverables')"&gt;
        &lt;div class="item-title"&gt;${escapeHtml(d.description)}&lt;/div&gt;
        &lt;div class="item-meta"&gt;
          ${etaDisplay}
          ${daysBadge}
        &lt;/div&gt;
      &lt;/div&gt;
    `;
  }).join('');
}

// Render page
function renderPage(page) {
  switch(page) {
    case 'overview':
      renderOverview();
      break;
    case 'schedule':
      renderSchedule();
      break;
    case 'deliverables':
      renderDeliverables();
      break;
```

INDEX.HTML (continued - lines 1041+)

```
      case 'documents':
        renderDocuments();
        break;
      case 'links':
        renderLinks();
        break;
      default:
        renderOverview();
    }
  }

  // Render Overview
  function renderOverview() {
    const allDeliverables = [...dashboardData.exec1Deliverables, ...dashboardData.exec2Deliverables];
    const criticalItems = allDeliverables.filter(d =&gt; d.urgency === 'critical' || d.urgency === 'overdue').length;
    const upcomingMeetings = dashboardData.meetings.filter(m =&gt; m.weekNumber === 1).length;
    const hotTopics = dashboardData.meetings.filter(m =&gt; m.isHotTopic).length;
    const documentCount = dashboardData.documentRepoLinks.length;

    document.getElementById('content-area').innerHTML = `
      &lt;div class="grid grid-3" style="margin-bottom: 24px;"&gt;
        &lt;div class="stat-card"&gt;
          &lt;div class="stat-value" style="color: ${criticalItems &gt; 0 ? 'var(--danger)' : 'var(--success)'}"&gt;
            ${criticalItems}
          &lt;/div&gt;
          &lt;div class="stat-label"&gt;Critical/Overdue Items&lt;/div&gt;
        &lt;/div&gt;
        &lt;div class="stat-card"&gt;
          &lt;div class="stat-value"&gt;${upcomingMeetings}&lt;/div&gt;
          &lt;div class="stat-label"&gt;Meetings This Week&lt;/div&gt;
        &lt;/div&gt;
        &lt;div class="stat-card"&gt;
          &lt;div class="stat-value" style="color: ${hotTopics &gt; 0 ? 'var(--verizon-red)' : 'inherit'}"&gt;
            ${hotTopics}
          &lt;/div&gt;
          &lt;div class="stat-label"&gt;Hot Topics&lt;/div&gt;
        &lt;/div&gt;
      &lt;/div&gt;

      &lt;div class="grid grid-2"&gt;
        &lt;div class="card"&gt;
          &lt;div class="card-header"&gt;
            &lt;h3 class="card-title"&gt;Upcoming Deliverables&lt;/h3&gt;
            &lt;button class="btn btn-secondary" onclick="navigateTo('deliverables')"&gt;View All&lt;/button&gt;
          &lt;/div&gt;
          &lt;div class="card-body"&gt;
            ${renderDeliverablesList(allDeliverables.slice(0, 5))}
          &lt;/div&gt;
        &lt;/div&gt;

        &lt;div class="card"&gt;
          &lt;div class="card-header"&gt;
            &lt;h3 class="card-title"&gt;Today's Meetings&lt;/h3&gt;
            &lt;button class="btn btn-secondary" onclick="navigateTo('schedule')"&gt;View Schedule&lt;/button&gt;
          &lt;/div&gt;
          &lt;div class="card-body"&gt;
            ${renderTodaysMeetings()}
          &lt;/div&gt;
        &lt;/div&gt;
      &lt;/div&gt;

      &lt;!-- Document Repository Section on Overview --&gt;
      &lt;div class="card doc-repo-section"&gt;
        &lt;div class="card-header"&gt;
          &lt;h3 class="card-title"&gt;Document Repository&lt;/h3&gt;
          &lt;button class="btn btn-secondary" onclick="navigateTo('documents')"&gt;View All&lt;/button&gt;
        &lt;/div&gt;
        &lt;div class="card-body"&gt;
          ${renderDocumentRepoList(dashboardData.documentRepoLinks.slice(0, 5))}
        &lt;/div&gt;
      &lt;/div&gt;
    `;
  }

  // Render Document Repository List (for overview and documents page)
  function renderDocumentRepoList(documents) {
    if (!documents || documents.length === 0) {
      return '&lt;div class="empty-state"&gt;&lt;p&gt;No documents available&lt;/p&gt;&lt;/div&gt;';
    }
```

INDEX.HTML (continued - lines 1121+)

```
  return `
    &lt;div class="doc-repo-list"&gt;
      ${documents.map(doc =&gt; `
        &lt;div class="doc-repo-item"&gt;
          &lt;i class="fas fa-file-alt"&gt;&lt;/i&gt;
          &lt;div class="doc-info"&gt;
            &lt;div class="doc-name"&gt;${escapeHtml(doc.description || doc.name)}&lt;/div&gt;
            &lt;div class="doc-meta"&gt;
              ${doc.owner ? `Owner: ${escapeHtml(doc.owner)}` : ''}
              ${doc.comment ? ` • ${escapeHtml(doc.comment)}` : ''}
            &lt;/div&gt;
          &lt;/div&gt;
          ${doc.url || doc.link ? `
            &lt;a href="${doc.url || doc.link}" target="_blank" class="doc-link"&gt;
              &lt;i class="fas fa-external-link-alt"&gt;&lt;/i&gt; Open
            &lt;/a&gt;
          ` : ''}
        &lt;/div&gt;
      `).join('')}
    &lt;/div&gt;
  `;
}

// Render deliverables list
function renderDeliverablesList(deliverables) {
  if (!deliverables || deliverables.length === 0) {
    return '&lt;div class="empty-state"&gt;&lt;p&gt;No deliverables&lt;/p&gt;&lt;/div&gt;';
  }

  return `
    &lt;table&gt;
      &lt;thead&gt;
        &lt;tr&gt;
          &lt;th&gt;Description&lt;/th&gt;
          &lt;th&gt;ETA&lt;/th&gt;
          &lt;th&gt;Days Left&lt;/th&gt;
          &lt;th&gt;Owner&lt;/th&gt;
        &lt;/tr&gt;
      &lt;/thead&gt;
      &lt;tbody&gt;
        ${deliverables.map(d =&gt; {
          // FIX: Handle text-based daysLeft values
          let daysLeftDisplay = '';
          if (d.daysRemainingText &amp;&amp; typeof d.daysRemainingText === 'string' &amp;&amp; !d.daysRemainingText.match(/^\d+d$/)) {
            // It's text like "weekly", "monthly" - display as-is with neutral badge
            daysLeftDisplay = `&lt;span class="badge badge-text"&gt;${escapeHtml(d.daysRemainingText)}&lt;/span&gt;`;
          } else if (d.daysRemaining !== null &amp;&amp; typeof d.daysRemaining === 'number') {
            // It's numeric - display with urgency badge
            daysLeftDisplay = `&lt;span class="badge badge-${d.urgency}"&gt;${Math.abs(d.daysRemaining)}d ${d.daysRemaining &lt; 0 ? 'overdue' : ''}
          }

          return `
            &lt;tr&gt;
              &lt;td&gt;${escapeHtml(d.description)}&lt;/td&gt;
              &lt;td&gt;${d.eta || '-'}&lt;/td&gt;
              &lt;td&gt;${daysLeftDisplay}&lt;/td&gt;
              &lt;td&gt;${escapeHtml(d.owner) || '-'}&lt;/td&gt;
            &lt;/tr&gt;
          `;
        }).join('')}
      &lt;/tbody&gt;
    &lt;/table&gt;
  `;
}

// Render today's meetings
function renderTodaysMeetings() {
  const today = new Date().toISOString().split('T')[0];
  const todaysMeetings = dashboardData.meetings.filter(m =&gt; m.date === today);

  if (todaysMeetings.length === 0) {
    return '&lt;div class="empty-state"&gt;&lt;i class="fas fa-calendar-check"&gt;&lt;/i&gt;&lt;p&gt;No meetings today&lt;/p&gt;&lt;/div&gt;';
  }

  return todaysMeetings.map(m =&gt; `
    &lt;div class="schedule-meeting ${m.isHotTopic ? 'hot-topic' : ''} ${m.prepRequired ? 'prep-required' : ''}"&gt;
      &lt;div class="meeting-time"&gt;${m.time || 'TBD'}&lt;/div&gt;
      &lt;div class="meeting-title"&gt;${escapeHtml(m.title)}&lt;/div&gt;
      &lt;div class="meeting-badges"&gt;
        ${m.isHotTopic ? '&lt;span class="meeting-badge hot"&gt;HOT&lt;/span&gt;' : ''}
```

INDEX.HTML (continued - lines 1201+)

```
        ${m.prepRequired ? '&lt;span class="meeting-badge prep"&gt;PREP&lt;/span&gt;' : ''}
      &lt;/div&gt;
    &lt;/div&gt;
  `).join('');
}

// Render Schedule
function renderSchedule() {
  const weekMeetings = dashboardData.meetings.filter(m =&gt; m.weekNumber === currentWeek);

  // Get week dates
  const today = new Date();
  const currentDay = today.getDay();
  const diff = currentDay === 0 ? -6 : 1 - currentDay;
  const monday = new Date(today);
  monday.setDate(today.getDate() + diff);

  if (currentWeek === 2) {
    monday.setDate(monday.getDate() + 7);
  }

  const days = [];
  for (let i = 0; i &lt; 5; i++) {
    const date = new Date(monday);
    date.setDate(monday.getDate() + i);
    days.push({
      name: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'][i],
      date: date.toISOString().split('T')[0],
      dateFormatted: date.toLocaleDateString('en-US', { month: 'short', day: 'numeric' })
    });
  }

  document.getElementById('content-area').innerHTML = `
    &lt;div class="week-tabs"&gt;
      &lt;button class="week-tab ${currentWeek === 1 ? 'active' : ''}" onclick="switchWeek(1)"&gt;This Week&lt;/button&gt;
      &lt;button class="week-tab ${currentWeek === 2 ? 'active' : ''}" onclick="switchWeek(2)"&gt;Next Week&lt;/button&gt;
    &lt;/div&gt;

    &lt;div class="schedule-grid"&gt;
      ${days.map(day =&gt; {
        const dayMeetings = weekMeetings
          .filter(m =&gt; m.date === day.date)
          .sort((a, b) =&gt; (a.time24 || '').localeCompare(b.time24 || ''));

        return `
          &lt;div class="schedule-day"&gt;
            &lt;div class="schedule-day-header"&gt;
              &lt;div class="day-name"&gt;${day.name}&lt;/div&gt;
              &lt;div class="day-date"&gt;${day.dateFormatted}&lt;/div&gt;
            &lt;/div&gt;
            &lt;div class="schedule-day-content"&gt;
              ${dayMeetings.length === 0 ?
                '&lt;div style="color: var(--gray-400); font-size: 13px; text-align: center; padding: 20px;"&gt;No meetings&lt;/div&gt;' :
                dayMeetings.map(m =&gt; `
                  &lt;div class="schedule-meeting ${m.isHotTopic ? 'hot-topic' : ''} ${m.prepRequired ? 'prep-required' : ''} ${m.hasConflict ? 'has
                    &lt;div class="meeting-time"&gt;${m.time || 'TBD'}&lt;/div&gt;
                    &lt;div class="meeting-title"&gt;${escapeHtml(m.title)}&lt;/div&gt;
                    &lt;div class="meeting-badges"&gt;
                      ${m.isHotTopic ? '&lt;span class="meeting-badge hot"&gt;HOT&lt;/span&gt;' : ''}
                      ${m.prepRequired ? '&lt;span class="meeting-badge prep"&gt;PREP&lt;/span&gt;' : ''}
                      ${m.hasConflict ? '&lt;span class="meeting-badge conflict"&gt;CONFLICT&lt;/span&gt;' : ''}
                    &lt;/div&gt;
                  &lt;/div&gt;
                `).join('')
              }
            &lt;/div&gt;
          &lt;/div&gt;
        `;
      }).join('')}
    &lt;/div&gt;
  `;
}

// Switch week
function switchWeek(week) {
  currentWeek = week;
  renderSchedule();
}

// Render Deliverables
```

INDEX.HTML (continued - lines 1281+)

```
function renderDeliverables() {
  document.getElementById('content-area').innerHTML = `
    &lt;div class="tabs"&gt;
      &lt;div class="tab active" data-tab="exec1"&gt;VP Deliverables&lt;/div&gt;
      &lt;div class="tab" data-tab="exec2"&gt;Governance Deliverables&lt;/div&gt;
      &lt;div class="tab" data-tab="completed"&gt;Completed&lt;/div&gt;
    &lt;/div&gt;
    &lt;div id="deliverables-content"&gt;
      ${renderDeliverablesTable(dashboardData.exec1Deliverables)}
    &lt;/div&gt;
  `;

  // Setup tab switching
  document.querySelectorAll('.tabs .tab').forEach(tab =&gt; {
    tab.addEventListener('click', () =&gt; {
      document.querySelectorAll('.tabs .tab').forEach(t =&gt; t.classList.remove('active'));
      tab.classList.add('active');

      const tabName = tab.dataset.tab;
      let data = [];
      if (tabName === 'exec1') data = dashboardData.exec1Deliverables;
      else if (tabName === 'exec2') data = dashboardData.exec2Deliverables;
      else if (tabName === 'completed') data = dashboardData.completedItems;

      document.getElementById('deliverables-content').innerHTML = renderDeliverablesTable(data);
    });
  });
}

// Render deliverables table
function renderDeliverablesTable(deliverables) {
  if (!deliverables || deliverables.length === 0) {
    return '&lt;div class="empty-state"&gt;&lt;i class="fas fa-check-circle"&gt;&lt;/i&gt;&lt;p&gt;No items&lt;/p&gt;&lt;/div&gt;';
  }

  return `
    &lt;div class="card"&gt;
      &lt;div class="table-container"&gt;
        &lt;table&gt;
          &lt;thead&gt;
            &lt;tr&gt;
              &lt;th&gt;Description&lt;/th&gt;
              &lt;th&gt;ETA&lt;/th&gt;
              &lt;th&gt;Days Left&lt;/th&gt;
              &lt;th&gt;Owner&lt;/th&gt;
              &lt;th&gt;Link&lt;/th&gt;
              &lt;th&gt;Comments&lt;/th&gt;
            &lt;/tr&gt;
          &lt;/thead&gt;
          &lt;tbody&gt;
            ${deliverables.map(d =&gt; {
              // FIX: Handle text-based daysLeft values (like "weekly", "monthly")
              let daysLeftDisplay = '';
              const rawDaysLeft = d.daysLeft || d.daysRemainingText || '';

              // Check if it's a text value (not numeric)
              if (typeof rawDaysLeft === 'string' &amp;&amp; rawDaysLeft &amp;&amp; !rawDaysLeft.match(/^-?\d+d?$/)) {
                // Text value like "weekly", "monthly" - display as-is
                daysLeftDisplay = `&lt;span class="badge badge-text"&gt;${escapeHtml(rawDaysLeft)}&lt;/span&gt;`;
              } else if (d.daysRemaining !== null &amp;&amp; d.daysRemaining !== undefined &amp;&amp; typeof d.daysRemaining === 'number') {
                // Numeric value - calculate and display with urgency
                daysLeftDisplay = `&lt;span class="badge badge-${d.urgency}"&gt;${Math.abs(d.daysRemaining)}d ${d.daysRemaining &lt; 0 ? 'overdue' :
              }

              return `
                &lt;tr&gt;
                  &lt;td&gt;${escapeHtml(d.description)}&lt;/td&gt;
                  &lt;td&gt;${d.eta || '-'}&lt;/td&gt;
                  &lt;td&gt;${daysLeftDisplay}&lt;/td&gt;
                  &lt;td&gt;${escapeHtml(d.owner) || '-'}&lt;/td&gt;
                  &lt;td&gt;
                    ${d.linkedMaterials || d.link ?
                      `&lt;a href="${d.linkedMaterials || d.link}" target="_blank" class="btn btn-secondary" style="padding: 4px 8px; font-size: 12p
                        &lt;i class="fas fa-external-link-alt"&gt;&lt;/i&gt;
                      &lt;/a&gt;` : '-'}
                  &lt;/td&gt;
                  &lt;td&gt;${escapeHtml(d.comment) || '-'}&lt;/td&gt;
                &lt;/tr&gt;
              `;
            }).join('')}
```

INDEX.HTML (continued - lines 1361+)

```
          &lt;/tbody&gt;
        &lt;/table&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  `;
}

// Render Documents page with correct columns: Description, Owner, Link to Materials, Comments
function renderDocuments() {
  const documents = dashboardData.documentRepoLinks || [];

  document.getElementById('content-area').innerHTML = `
    &lt;div class="card"&gt;
      &lt;div class="card-header"&gt;
        &lt;h3 class="card-title"&gt;Document Repository&lt;/h3&gt;
        &lt;span style="color: var(--gray-500); font-size: 14px;"&gt;${documents.length} documents&lt;/span&gt;
      &lt;/div&gt;
      &lt;div class="table-container"&gt;
        &lt;table&gt;
          &lt;thead&gt;
            &lt;tr&gt;
              &lt;th&gt;Description&lt;/th&gt;
              &lt;th&gt;Owner&lt;/th&gt;
              &lt;th&gt;Link to Materials&lt;/th&gt;
              &lt;th&gt;Comments&lt;/th&gt;
            &lt;/tr&gt;
          &lt;/thead&gt;
          &lt;tbody&gt;
            ${documents.length === 0 ?
              `&lt;tr&gt;&lt;td colspan="4" style="text-align: center; padding: 40px; color: var(--gray-500);"&gt;
                &lt;i class="fas fa-folder-open" style="font-size: 32px; margin-bottom: 12px; display: block;"&gt;&lt;/i&gt;
                No documents in repository
              &lt;/td&gt;&lt;/tr&gt;` :
              documents.map(doc =&gt; `
                &lt;tr&gt;
                  &lt;td&gt;
                    &lt;div style="display: flex; align-items: center; gap: 10px;"&gt;
                      &lt;i class="fas fa-file-alt" style="color: var(--verizon-red);"&gt;&lt;/i&gt;
                      &lt;span&gt;${escapeHtml(doc.description || doc.name)}&lt;/span&gt;
                    &lt;/div&gt;
                  &lt;/td&gt;
                  &lt;td&gt;${escapeHtml(doc.owner) || '-'}&lt;/td&gt;
                  &lt;td&gt;
                    ${doc.url || doc.link ?
                      `&lt;a href="${doc.url || doc.link}" target="_blank" class="btn btn-secondary" style="padding: 4px 12px; font-size: 12px;"&gt;
                        &lt;i class="fas fa-external-link-alt"&gt;&lt;/i&gt; Open
                      &lt;/a&gt;` :
                      '&lt;span style="color: var(--gray-400);"&gt;No link&lt;/span&gt;'}
                  &lt;/td&gt;
                  &lt;td&gt;${escapeHtml(doc.comment) || '-'}&lt;/td&gt;
                &lt;/tr&gt;
              `).join('')
            }
          &lt;/tbody&gt;
        &lt;/table&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  `;
}

// Render Links
function renderLinks() {
  document.getElementById('content-area').innerHTML = `
    &lt;div class="grid grid-2"&gt;
      &lt;div class="card"&gt;
        &lt;div class="card-header"&gt;
          &lt;h3 class="card-title"&gt;Admin &amp; Settings&lt;/h3&gt;
        &lt;/div&gt;
        &lt;div class="card-body"&gt;
          &lt;div class="links-grid"&gt;
            ${dashboardData.adminLinks.map(link =&gt; `
              &lt;a href="${link.url}" target="_blank" class="link-card"&gt;
                &lt;i class="fas fa-cog"&gt;&lt;/i&gt;
                &lt;span&gt;${escapeHtml(link.name)}&lt;/span&gt;
              &lt;/a&gt;
            `).join('')}
          &lt;/div&gt;
        &lt;/div&gt;
      &lt;/div&gt;
```

INDEX.HTML (continued - lines 1441+)

```
      &lt;div class="card"&gt;
        &lt;div class="card-header"&gt;
          &lt;h3 class="card-title"&gt;KPIs &amp; Dashboards&lt;/h3&gt;
        &lt;/div&gt;
        &lt;div class="card-body"&gt;
          &lt;div class="links-grid"&gt;
            ${dashboardData.kpiLinks.map(link =&gt; `
              &lt;a href="${link.url}" target="_blank" class="link-card"&gt;
                &lt;i class="fas fa-chart-line"&gt;&lt;/i&gt;
                &lt;span&gt;${escapeHtml(link.name)}&lt;/span&gt;
              &lt;/a&gt;
            `).join('')}
          &lt;/div&gt;
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;

    &lt;div class="card" style="margin-top: 24px;"&gt;
      &lt;div class="card-header"&gt;
        &lt;h3 class="card-title"&gt;Executive Updates Archive&lt;/h3&gt;
      &lt;/div&gt;
      &lt;div class="card-body"&gt;
        &lt;div class="links-grid"&gt;
          ${dashboardData.executiveUpdates.map(update =&gt; `
            &lt;a href="${update.url}" target="_blank" class="link-card"&gt;
              &lt;i class="fas fa-file-alt"&gt;&lt;/i&gt;
              &lt;span&gt;${escapeHtml(update.name)}&lt;/span&gt;
            &lt;/a&gt;
          `).join('')}
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  `;
}

// PDF Export Modal
function openPDFExportModal() {
  const modal = document.getElementById('pdf-modal');
  const body = document.getElementById('pdf-modal-body');

  body.innerHTML = `
    &lt;div style="margin-bottom: 20px;"&gt;
      &lt;label style="display: block; margin-bottom: 8px; font-weight: 500;"&gt;Week Title&lt;/label&gt;
      &lt;input type="text" id="pdf-title" class="form-control" value="The Current Week: A Funnel from Data to Decision"
        style="width: 100%; padding: 10px; border: 1px solid var(--gray-300); border-radius: 6px;"&gt;
    &lt;/div&gt;

    &lt;div style="margin-bottom: 20px;"&gt;
      &lt;label style="display: flex; align-items: center; gap: 8px;"&gt;
        &lt;input type="checkbox" id="pdf-prep" checked&gt;
        &lt;span&gt;Include prep meeting indicators&lt;/span&gt;
      &lt;/label&gt;
    &lt;/div&gt;

    &lt;div&gt;
      &lt;label style="display: block; margin-bottom: 8px; font-weight: 500;"&gt;Select Meetings to Include&lt;/label&gt;
      &lt;div style="max-height: 300px; overflow-y: auto; border: 1px solid var(--gray-200); border-radius: 6px; padding: 12px;"&gt;
        ${dashboardData.meetings.map(m =&gt; `
          &lt;label style="display: flex; align-items: flex-start; gap: 8px; padding: 8px 0; border-bottom: 1px solid var(--gray-100);"&gt;
            &lt;input type="checkbox" class="meeting-checkbox" value="${m.id}" checked&gt;
            &lt;div&gt;
              &lt;div style="font-weight: 500;"&gt;${escapeHtml(m.title)}&lt;/div&gt;
              &lt;div style="font-size: 12px; color: var(--gray-500);"&gt;
                ${m.dateFormatted} at ${m.time}
                ${m.hasConflict ? '&lt;span style="color: #ff6600;"&gt; ■ Conflict&lt;/span&gt;' : ''}
              &lt;/div&gt;
            &lt;/div&gt;
          &lt;/label&gt;
        `).join('')}
      &lt;/div&gt;
    &lt;/div&gt;
  `;

  modal.classList.add('active');
}

function closePDFModal() {
  document.getElementById('pdf-modal').classList.remove('active');
}
```

INDEX.HTML (continued - lines 1521+)

```
    function generatePDF() {
      const title = document.getElementById('pdf-title').value;
      const includePrepMeetings = document.getElementById('pdf-prep').checked;
      const selectedMeetings = Array.from(document.querySelectorAll('.meeting-checkbox:checked')).map(cb =&gt; cb.value);

      // Show loading
      document.getElementById('pdf-modal-body').innerHTML = `
        &lt;div class="loading"&gt;
          &lt;i class="fas fa-spinner"&gt;&lt;/i&gt; Generating PDF...
        &lt;/div&gt;
      `;

      google.script.run
        .withSuccessHandler(result =&gt; {
          if (result.success) {
            window.open(result.downloadUrl, '_blank');
            closePDFModal();
          } else {
            alert('Error generating PDF: ' + result.error);
            openPDFExportModal();
          }
        })
        .withFailureHandler(error =&gt; {
          alert('Error: ' + error);
          openPDFExportModal();
        })
        .generateWeeklySchedulePDF({
          weekTitle: title,
          includePrepMeetings: includePrepMeetings,
          selectedMeetingIds: selectedMeetings
        });
    }

    // Utility: Escape HTML
    function escapeHtml(text) {
      if (!text) return '';
      const div = document.createElement('div');
      div.textContent = text;
      return div.innerHTML;
    }
  &lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
```