Appendix 1 - Header and Screen Help Management - Code

This is the code for the header of all screens. Within this is the js scripts that control the model for the help screen display.

```
function writePageHeader($showUser = false) {
  global $pctCompanyName, $pctLoggedOnUserID, $screenName, $screenTitle;

  $user    = ($showUser && !empty($pctLoggedOnUserID)) ? $pctLoggedOnUserID : "---";
  $company = $pctCompanyName ?? "Your Company";
  $screen  = $screenName ?? "Unknown";

  if ($screen === "index.php") {
    $screen = "Logon";
  }

  $rowhtml = "";

  ############################################################
  # Banner (3 columns: Left brand, Center actions, Right screen info)
  ############################################################
  $rowhtml  = "<div style='border:2px solid #2596be; background:#eaf6fb; padding:10px 12px; margin:20px auto; width:80%; border-radius:12px;'>";
  $rowhtml .= "  <div style='display:flex; justify-content:space-between; align-items:center; flex-wrap:nowrap; gap:14px;'>";
```

```
   ############################################################

   # LEFT COLUMN: combined brand badge (logo + system name)

   ############################################################

   $rowhtml .= "    <div style='flex:0 0 320px; display:flex; align-items:center; justify-content:flex-start;'>";

   $rowhtml .= "     <div style='display:flex; align-items:center; gap:12px; background:#e3f2fd; border:1px solid #cfe7f3; border-radius:12px; padding:10px 14px;'>";

   $rowhtml .= "       <div style='flex:0 0 72px; height:72px; display:flex; align-items:center; justify-content:center; overflow:hidden;'>";

   $rowhtml .= "        <img src='/images/BookBrandingLogo.png' alt='Logo' style='max-height:72px; max-width:72px; width:auto; height:auto; object-fit:contain;'>";

   $rowhtml .= "       </div>";

   $rowhtml .= "       <div style='flex:1 1 auto; font-size:20px; font-weight:600; color:#2596be; line-height:1.1; white-space:nowrap;'>Corporate Intelligence System</div>";

   $rowhtml .= "     </div>";

   $rowhtml .= "    </div>";


   ############################################################

   # CENTER COLUMN: 3 lines

   # 1) Home + Screen Help + Help(?)

   # 2) Search Manual for Help + Search icon

   # 3) Chat with Me + chat icon
```

```
   ############################################################
   $rowhtml .= "    <div style='flex:1 1 auto; display:flex; flex-direction:column; justify-content:center; align-items:center; gap:8px; min-width:260px; min-width:0;'>";

   ############################################################
   # Center Line 1: Home + Screen Help + Help icon
   ############################################################
   $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center; gap:10px; flex-wrap:nowrap;'>";

   if (strtolower($screen) !== "logon" && strtolower($screenName ?? "") !== "logon") {
      $rowhtml .= "       <a href='Menu_Main.php' ";
      $rowhtml .= "         style='display:inline-block; padding:4px 12px; background-color:#2596be; color:white; text-decoration:none; border-radius:6px; font-size:13px;'>Home</a>";
   }

   $rowhtml .= "       <span style='font-size:14px; color:#2596be; font-weight:bold;'>Screen Help</span>";

   $rowhtml .= "       <img src='/images/questionmark.png' alt='Help' title='Screen Help' ";
   $rowhtml .= "          style='cursor:pointer; width:18px; height:18px;' ";
   $rowhtml .= "          onclick=\"showScreenHelp('" . htmlspecialchars($screen, ENT_QUOTES, 'UTF-8') . "')\">";
```

```
  $rowhtml .= "     </div>";


  ###############################################################

  # Center Line 2: Search box + Search icon

  ###############################################################

  $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center;
gap:6px; width:100%;'>";

  $rowhtml .= "      <input type='text' name='txt_Question' id='txt_Question'
placeholder='Search Manual for Help' ";

  $rowhtml .= "           style='max-width:520px; width:75%; padding:5px 10px; border:1px
solid #ccc; border-radius:6px; font-size:14px;' ";

  $rowhtml .= "           onkeydown=\"if(event.key==='Enter'){submitHeaderQuestion();}\">";

  $rowhtml .= "      <img src='/images/Search.png' alt='Search' title='Search Help' ";

  $rowhtml .= "          style='cursor:pointer; width:18px; height:18px; vertical-align:middle;'
";

  $rowhtml .= "          onclick='submitHeaderQuestion()'>";

  $rowhtml .= "     </div>";


  ###############################################################

  # Center Line 3: Chat with Me + chat icon

  ###############################################################

  $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center;
gap:8px;'>";
```

```
    $rowhtml .= "       <a href='Chat.php' target='_blank' style='display:flex; align-items:center; gap:8px; text-decoration:none;'>";
    $rowhtml .= "        <span style='font-size:14px; color:#2596be; font-weight:bold;'>Chat with Me</span>";
    $rowhtml .= "        <img src='/images/chat.png' alt='Chat' title='Chat' ";
    $rowhtml .= "            style='width:26px; height:26px; vertical-align:middle;'>";
    $rowhtml .= "       </a>";
    $rowhtml .= "     </div>";
    $rowhtml .= "    </div>";

    ############################################################
    # RIGHT COLUMN: 2 lines with readonly inputs (no postback)
    # 1) Screen Name
    # 2) Description
    ############################################################
    $rowhtml .= "    <div style='flex:0 0 340px; display:flex; flex-direction:column; justify-content:center; gap:8px;'>";

    $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:space-between; gap:8px;'>";
    $rowhtml .= "       <div style='color:#2596be; font-size:13px; font-weight:bold; white-space:nowrap;'>Screen Name</div>";
    $rowhtml .= "       <input type='text' readonly='readonly' value='" . htmlspecialchars($screenName ?? "", ENT_QUOTES, 'UTF-8') . "' ";
```

```
    $rowhtml .= "            style='width:220px; padding:4px 8px; border:1px solid #ccc; border-radius:6px; font-size:13px; background:#f7fbfe;'>";
    $rowhtml .= "     </div>";

    $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:space-between; gap:8px;'>";
    $rowhtml .= "       <div style='color:#2596be; font-size:13px; font-weight:bold; white-space:nowrap;'>Description</div>";
    $rowhtml .= "       <input type='text' readonly='readonly' value='" . htmlspecialchars($screenTitle ?? "", ENT_QUOTES, 'UTF-8') . "' ";
    $rowhtml .= "            style='width:220px; padding:4px 8px; border:1px solid #ccc; border-radius:6px; font-size:13px; background:#f7fbfe;'>";
    $rowhtml .= "     </div>";

    $rowhtml .= "    </div>";

    ############################################################
    # Close banner containers
    ############################################################
    $rowhtml .= "  </div>";
    $rowhtml .= "</div>";

    echo $rowhtml;
```

```
    ############################################################
    # Help modal + scripts
    ############################################################
    echo <<<EOT
<script>
function submitHeaderQuestion() {
    var el = document.getElementById('txt_Question');
    if (!el) return;

    var q = (el.value || '').trim();
    if (q.length === 0) { return; }

    var url = 'Search.php?Question=' + encodeURIComponent(q);
    var win = window.open(url, '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}

function openHeaderChat() {
    var win = window.open('Chat.php', '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}
```

```
function showScreenHelp(screen) {
   console.log("Screen requested:", screen);

   var modal = document.getElementById('helpModal');
   var body  = document.getElementById('helpModalBody');
   if (!modal || !body) return;

   body.innerHTML = "<div style='font-size:12px;color:#666;'>Loading...</div>";

   fetch('gethelp.php?screen=' + encodeURIComponent(screen))
      .then(function(response){ return response.JSON(); })
      .then(function(data){
         if (!data || data.status !== 'ok') {
            var msg = (data && data.message) ? data.message : 'No help available for this
screen.';
            body.innerHTML = escapeHtml(msg);
            modal.style.display = 'block';
            return;
         }

         var mode        = data.HelpMode || 'Text';
         var chapterName  = data.ChapterName || '';
         var sectionName  = data.SectionName || '';
```

```
var showHeader   = (data.ShowHeader === 'Y');
var content      = data.Content || '';

var html = "";

if (mode === 'HTML') {
   if (showHeader && (chapterName || sectionName)) {
      html += "<div style='margin:0 0 8px 0;'><strong>" +
            escapeHtml(chapterName) + " - " + escapeHtml(sectionName) +
            "</strong></div>";
   }
   html += content;
} else {
   html += escapeHtml(content).replace(/\\n/g, "<br>");
}

body.innerHTML = html;

var imgs = body.querySelectorAll('img');
imgs.forEach(function(img){
   img.style.maxWidth = "100%";
   img.style.height   = "auto";
```

```
            img.style.cursor   = "zoom-in";
            img.addEventListener('click', function(){
               var src = img.getAttribute('src');
               if (src) {
                  var win = window.open(src, '_blank', 'noopener,noreferrer');
                  if (win) { win.opener = null; }
               }
            });
         });

         modal.style.display = 'block';
      })
      .catch(function(error){
         console.error("Fetch Error:", error);
         body.innerHTML = "Error loading help.";
         modal.style.display = 'block';
      });
}

function closeHelpModal() {
   document.getElementById('helpModal').style.display = 'none';
}
```

```
function escapeHtml(s) {
  s = String(s || '');
  return s
    .replace(/&/g, "&")
    .replace(/</g, "<")
    .replace(/>/g, ">")
    .replace(/"/g, """)
    .replace(/'/g, "'");
}
</script>

<div id="helpModal" style="display:none; position:fixed; z-index:9999; left:0; top:0; width:100%; height:100%; overflow:auto; background-color:rgba(0,0,0,0.4);">
 <div style="background-color:#fff; margin:10% auto; padding:20px; border:1px solid #888; width:50%; border-radius:10px;">
  <span onclick="closeHelpModal()" style="color:#aaa; float:right; font-size:28px; font-weight:bold; cursor:pointer;">×</span>
  <h4>Screen Help</h4>
  <div id="helpModalBody" style="margin-top:10px;">
  </div>
 </div>
</div>
```

```
EOT;

}
```

Appendix 2 - Table Driven Menu Bar - Code

The following represents in php code the table driven menu bar displayed in all GUI front ends.

```
function writePageHeader($showUser = false) {
  global $pctCompanyName, $pctLoggedOnUserID, $screenName, $screenTitle;

  $user    = ($showUser && !empty($pctLoggedOnUserID)) ? $pctLoggedOnUserID : "---";
  $company = $pctCompanyName ?? "Your Company";
  $screen  = $screenName ?? "Unknown";

  if ($screen === "index.php") {
    $screen = "Logon";
  }

  $rowhtml = "";

  ############################################################
  # Banner (3 columns: Left brand, Center actions, Right screen info)
  ############################################################
  $rowhtml  = "<div style='border:2px solid #2596be; background:#eaf6fb; padding:10px
12px; margin:20px auto; width:80%; border-radius:12px;'>";
```

```
$rowhtml .= "  <div style='display:flex; justify-content:space-between; align-items:center; flex-wrap:nowrap; gap:14px;'>";

#############################################################

# LEFT COLUMN: combined brand badge (logo + system name)

#############################################################

$rowhtml .= "    <div style='flex:0 0 320px; display:flex; align-items:center; justify-content:flex-start;'>";

$rowhtml .= "     <div style='display:flex; align-items:center; gap:12px; background:#e3f2fd; border:1px solid #cfe7f3; border-radius:12px; padding:10px 14px;'>";

$rowhtml .= "       <div style='flex:0 0 72px; height:72px; display:flex; align-items:center; justify-content:center; overflow:hidden;'>";

$rowhtml .= "        <img src='/images/BookBrandingLogo.png' alt='Logo' style='max-height:72px; max-width:72px; width:auto; height:auto; object-fit:contain;'>";

$rowhtml .= "       </div>";

$rowhtml .= "       <div style='flex:1 1 auto; font-size:20px; font-weight:600; color:#2596be; line-height:1.1; white-space:nowrap;'>Corporate Intelligence System</div>";

$rowhtml .= "     </div>";

$rowhtml .= "    </div>";

#############################################################

# CENTER COLUMN: 3 lines

# 1) Home + Screen Help + Help(?)
```

```
  # 2) Search Manual for Help + Search icon
  # 3) Chat with Me + chat icon
  ##############################################################
  $rowhtml .= "    <div style='flex:1 1 auto; display:flex; flex-direction:column; justify-content:center; align-items:center; gap:8px; min-width:260px; min-width:0;'>";

  ##############################################################
  # Center Line 1: Home + Screen Help + Help icon
  ##############################################################
  $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center; gap:10px; flex-wrap:nowrap;'>";

  if (strtolower($screen) !== "logon" && strtolower($screenName ?? "") !== "logon") {
    $rowhtml .= "       <a href='Menu_Main.php' ";
    $rowhtml .= "         style='display:inline-block; padding:4px 12px; background-color:#2596be; color:white; text-decoration:none; border-radius:6px; font-size:13px;'>Home</a>";
  }

  $rowhtml .= "       <span style='font-size:14px; color:#2596be; font-weight:bold;'>Screen Help</span>";

  $rowhtml .= "       <img src='/images/questionmark.png' alt='Help' title='Screen Help' ";
  $rowhtml .= "          style='cursor:pointer; width:18px; height:18px;' ";
```

```
  $rowhtml .= "          onclick=\"showScreenHelp('" . htmlspecialchars($screen, ENT_QUOTES, 'UTF-8') . "')\">";
  $rowhtml .= "     </div>";

  ###############################################################
  # Center Line 2: Search box + Search icon
  ###############################################################
  $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center; gap:6px; width:100%;'>";
  $rowhtml .= "       <input type='text' name='txt_Question' id='txt_Question' placeholder='Search Manual for Help' ";
  $rowhtml .= "            style='max-width:520px; width:75%; padding:5px 10px; border:1px solid #ccc; border-radius:6px; font-size:14px;' ";
  $rowhtml .= "            onkeydown=\"if(event.key==='Enter'){submitHeaderQuestion();}\">";
  $rowhtml .= "       <img src='/images/Search.png' alt='Search' title='Search Help' ";
  $rowhtml .= "          style='cursor:pointer; width:18px; height:18px; vertical-align:middle;' ";
  $rowhtml .= "          onclick='submitHeaderQuestion()'>";
  $rowhtml .= "     </div>";

  ###############################################################
  # Center Line 3: Chat with Me + chat icon
  ###############################################################
```

```
$rowhtml .= "     <div style='display:flex; align-items:center; justify-content:center; gap:8px;'>";
$rowhtml .= "       <a href='Chat.php' target='_blank' style='display:flex; align-items:center; gap:8px; text-decoration:none;'>";
$rowhtml .= "        <span style='font-size:14px; color:#2596be; font-weight:bold;'>Chat with Me</span>";
$rowhtml .= "        <img src='/images/chat.png' alt='Chat' title='Chat' ";
$rowhtml .= "            style='width:26px; height:26px; vertical-align:middle;'>";
$rowhtml .= "       </a>";
$rowhtml .= "     </div>";
$rowhtml .= "   </div>";

   ############################################################
   # RIGHT COLUMN: 2 lines with readonly inputs (no postback)
   # 1) Screen Name
   # 2) Description
   ############################################################
$rowhtml .= "   <div style='flex:0 0 340px; display:flex; flex-direction:column; justify-content:center; gap:8px;'>";

$rowhtml .= "     <div style='display:flex; align-items:center; justify-content:space-between; gap:8px;'>";
$rowhtml .= "       <div style='color:#2596be; font-size:13px; font-weight:bold; white-space:nowrap;'>Screen Name</div>";
```

```
    $rowhtml .= "       <input type='text' readonly='readonly' value='" .
htmlspecialchars($screenName ?? "", ENT_QUOTES, 'UTF-8') . "' ";

    $rowhtml .= "            style='width:220px; padding:4px 8px; border:1px solid #ccc; border-
radius:6px; font-size:13px; background:#f7fbfe;'>";

    $rowhtml .= "     </div>";


    $rowhtml .= "     <div style='display:flex; align-items:center; justify-content:space-
between; gap:8px;'>";

    $rowhtml .= "       <div style='color:#2596be; font-size:13px; font-weight:bold; white-
space:nowrap;'>Description</div>";

    $rowhtml .= "       <input type='text' readonly='readonly' value='" .
htmlspecialchars($screenTitle ?? "", ENT_QUOTES, 'UTF-8') . "' ";

    $rowhtml .= "            style='width:220px; padding:4px 8px; border:1px solid #ccc; border-
radius:6px; font-size:13px; background:#f7fbfe;'>";

    $rowhtml .= "     </div>";


    $rowhtml .= "    </div>";


    ##############################################################

    # Close banner containers

    ##############################################################

    $rowhtml .= "  </div>";

    $rowhtml .= "</div>";
```

```
    echo $rowhtml;

    ################################################################
    # Help modal + scripts
    ################################################################
    echo <<<EOT
<script>
function submitHeaderQuestion() {
    var el = document.getElementById('txt_Question');
    if (!el) return;

    var q = (el.value || '').trim();
    if (q.length === 0) { return; }

    var url = 'Search.php?Question=' + encodeURIComponent(q);
    var win = window.open(url, '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
}

function openHeaderChat() {
    var win = window.open('Chat.php', '_blank', 'noopener,noreferrer');
    if (win) { win.opener = null; }
```

```
}

function showScreenHelp(screen) {
   console.log("Screen requested:", screen);

   var modal = document.getElementById('helpModal');
   var body  = document.getElementById('helpModalBody');
   if (!modal || !body) return;

   body.innerHTML = "<div style='font-size:12px;color:#666;'>Loading...</div>";

   fetch('gethelp.php?screen=' + encodeURIComponent(screen))
      .then(function(response){ return response.JSON(); })
      .then(function(data){
         if (!data || data.status !== 'ok') {
            var msg = (data && data.message) ? data.message : 'No help available for this
screen.';
            body.innerHTML = escapeHtml(msg);
            modal.style.display = 'block';
            return;
         }

         var mode       = data.HelpMode || 'Text';
```

```
var chapterName  = data.ChapterName || '';
var sectionName  = data.SectionName || '';
var showHeader   = (data.ShowHeader === 'Y');
var content      = data.Content || '';

var html = "";

if (mode === 'HTML') {
   if (showHeader && (chapterName || sectionName)) {
      html += "<div style='margin:0 0 8px 0;'><strong>" +
            escapeHtml(chapterName) + " - " + escapeHtml(sectionName) +
            "</strong></div>";
   }
   html += content;
} else {
   html += escapeHtml(content).replace(/\\n/g, "<br>");
}

body.innerHTML = html;

var imgs = body.querySelectorAll('img');
imgs.forEach(function(img){
```

```
            img.style.maxWidth = "100%";
            img.style.height   = "auto";
            img.style.cursor   = "zoom-in";
            img.addEventListener('click', function(){
               var src = img.getAttribute('src');
               if (src) {
                  var win = window.open(src, '_blank', 'noopener,noreferrer');
                  if (win) { win.opener = null; }
               }
            });
         });

         modal.style.display = 'block';
      })
      .catch(function(error){
         console.error("Fetch Error:", error);
         body.innerHTML = "Error loading help.";
         modal.style.display = 'block';
      });
}

function closeHelpModal() {
```

```
  document.getElementById('helpModal').style.display = 'none';
}

function escapeHtml(s) {
  s = String(s || '');
  return s
    .replace(/&/g, "&")
    .replace(/</g, "<")
    .replace(/>/g, ">")
    .replace(/"/g, """)
    .replace(/'/g, "'");
}
</script>

<div id="helpModal" style="display:none; position:fixed; z-index:9999; left:0; top:0; width:100%; height:100%; overflow:auto; background-color:rgba(0,0,0,0.4);">
 <div style="background-color:#fff; margin:10% auto; padding:20px; border:1px solid #888; width:50%; border-radius:10px;">
  <span onclick="closeHelpModal()" style="color:#aaa; float:right; font-size:28px; font-weight:bold; cursor:pointer;">×</span>
  <h4>Screen Help</h4>
  <div id="helpModalBody" style="margin-top:10px;">
  </div>
```

```
  </div>
</div>
EOT;
}
```

Appendix 3 - Make your Programs Domain Agnostic - Code

The following code provided in php to show functionality that can be converted to the programming language of your choice. These functions pick up and assign variables in order to avoid hardcoding domain names cpanel names and file high level direcotries for internal access and external file download as well as configuration of database signon email and other global configurations settings.

```
/*

=== Document Identity ===

DocType: Program

Category: Runtime Configuration Loader

Subsystem: Platform Core Website-Agnostic Configuration Management

programName: Runtime Configuration Functions

Audience: Both

Overview: Reads configuration files that make the application portable across domains, cPanel accounts, databases, email systems, filesystem paths, and AI providers.

Keywords: configuration, domain agnostic, runtime setup, filesystem, database, email, ai

=== End Document Identity ===

*/


/*

=== Program Purpose ===

This module loads all environment-specific values required by the application.
```

The same code base can be moved to another website by changing secure text

```
configuration files instead of editing program code.

=== End Program Purpose ===

*/


/*

=== loadRuntimeConfig() ===

Master loader for all runtime configuration sections.

*/

function loadRuntimeConfig()

{

   if (!setFileConfig()) { fatalConfigError("Unable to load file configuration."); }

   if (!loadDatabaseConfig()) { fatalConfigError("Unable to load database configuration."); }

   if (!loadSystemConfig()) { fatalConfigError("Unable to load system configuration."); }

   if (!loadEmailConfig()) { fatalConfigError("Unable to load email configuration."); }

   if (!loadAIConfig()) { fatalConfigError("Unable to load AI configuration."); }


   return true;

}


/*

=== fatalConfigError() ===

Displays a browser or command-line configuration error and stops execution.
```

```
*/
function fatalConfigError($message)
{
   $isWeb = (isset($_SERVER['HTTP_HOST']) || isset($_SERVER['REQUEST_METHOD']));

   if ($isWeb) {
      if (function_exists('ob_get_length')) {
         if (ob_get_length()) { ob_clean(); }
      }

      echo "<div style='margin:30px auto; max-width:900px; border:2px solid #dc3545; background:#fff1f1; padding:18px; font-family:Arial, Helvetica, sans-serif; border-radius:8px;'>";
      echo "<h3 style='color:#dc3545; margin-top:0;'>System Configuration Error</h3>";
      echo "<p style='font-size:15px; color:#111;'>" . htmlspecialchars((string)$message, ENT_QUOTES, 'UTF-8') . "</p>";
      echo "</div>";
   } else {
      echo "SYSTEM CONFIGURATION ERROR: " . $message . PHP_EOL;
   }

   exit;
}
```

```
/*
=== setFileConfig() ===
Derives the cPanel ID, domain folder, physical root, and URL base from __DIR__.
Stores the results in globals and session variables.
*/
function setFileConfig()
{
   global $cpanelid, $domain, $EnvironmentMode;
   global $FilePhysicalBase, $FileUrlBase;

   $cpanelid = '';
   $domain = '';
   $EnvironmentMode = 'legacy';
   $FilePhysicalBase = '';
   $FileUrlBase = '';

   $path = rtrim(str_replace('\\', '/', __DIR__), '/');

   $parts = explode('/', $path);
   $parts = array_values(array_filter($parts, 'strlen'));
```

```
if (count($parts) < 4) {
    fatalConfigError("Unable to determine website path from __DIR__: " . __DIR__);
}

if (strtolower($parts[0]) !== 'home') {
    fatalConfigError("Unexpected server path format: " . __DIR__);
}

$cpanelid = $parts[1] ?? '';
$siteFolder = $parts[2] ?? '';

if ($cpanelid === '') {
    fatalConfigError("Unable to determine cpanelid from path: " . __DIR__);
}

if ($siteFolder === '') {
    fatalConfigError("Unable to determine domain folder from path: " . __DIR__);
}

if (strpos($siteFolder, '.') !== false) {
    $tmp = explode('.', $siteFolder);
    $domain = $tmp[0];
```

```
    } else {
        $domain = $siteFolder;
    }

    $FilePhysicalBase = '/home/' . $cpanelid . '/' . $siteFolder;

    if (isset($parts[3]) && strtolower($parts[3]) === 'public_html') {
        $FilePhysicalBase .= '/public_html';
    }

    $FileUrlBase = 'https://' . $domain;

    if (isset($_SESSION)) {
        $_SESSION['cpanelid'] = $cpanelid;
        $_SESSION['domain'] = $domain;
        $_SESSION['FilePhysicalBase'] = $FilePhysicalBase;
        $_SESSION['FileUrlBase'] = $FileUrlBase;
        $_SESSION['EnvironmentMode'] = $EnvironmentMode;
    }

    return true;
}
```

```
/*
=== loadDatabaseConfig() ===
Reads secure/database.txt and loads operational and company database settings.
Missing operational values fall back to company values and vice versa.
Set up for twin database access
*/
function loadDatabaseConfig($file=null)
{
   global $EnvironmentMode;
   global $DB_SERVER,$DB_USER,$DB_PASSWORD,$DB_NAME,$DB_PORT;
   global $CDB_SERVER,$CDB_USER,$CDB_PASSWORD,$CDB_NAME,$CDB_PORT;
   global $FilePhysicalBase;

   $fileRoot = rtrim((string)$FilePhysicalBase, '/\\');
   if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadDatabaseConfig()."); }

   $defaultDir = $fileRoot . '/secure';

   if ($file === null || trim((string)$file) === '') {
      $file = $defaultDir . '/database.txt';
   } else {
```

```
    if (strpos($file, '/') === false && strpos($file, '\\') === false) {
        $file = $defaultDir . '/' . $file;
    }
}

if (!file_exists($file)) { fatalConfigError("Missing database config file: " . $file); }

$lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
if ($lines === false) { fatalConfigError("Unable to read database config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln, '=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));

    switch ($k) {
        case 'EnvironmentMode': $EnvironmentMode = $v; break;

        case 'DB_SERVER': $DB_SERVER = $v; break;
```

```
      case 'DB_USER': $DB_USER = $v; break;
      case 'DB_PASSWORD': $DB_PASSWORD = $v; break;
      case 'DB_NAME': $DB_NAME = $v; break;
      case 'DB_PORT': $DB_PORT = $v; break;

      case 'CDB_SERVER': $CDB_SERVER = $v; break;
      case 'CDB_USER': $CDB_USER = $v; break;
      case 'CDB_PASSWORD': $CDB_PASSWORD = $v; break;
      case 'CDB_NAME': $CDB_NAME = $v; break;
      case 'CDB_PORT': $CDB_PORT = $v; break;
    }
}

if (!$DB_SERVER) { $DB_SERVER = $CDB_SERVER; }
if (!$DB_USER) { $DB_USER = $CDB_USER; }
if (!$DB_PASSWORD) { $DB_PASSWORD = $CDB_PASSWORD; }
if (!$DB_NAME) { $DB_NAME = $CDB_NAME; }
if (!$DB_PORT) { $DB_PORT = $CDB_PORT; }

if (!$CDB_SERVER) { $CDB_SERVER = $DB_SERVER; }
if (!$CDB_USER) { $CDB_USER = $DB_USER; }
if (!$CDB_PASSWORD) { $CDB_PASSWORD = $DB_PASSWORD; }
```

```
    if (!$CDB_NAME) { $CDB_NAME = $DB_NAME; }
    if (!$CDB_PORT) { $CDB_PORT = $DB_PORT; }

    if (isset($_SESSION)) {
        $_SESSION['EnvironmentMode'] = $EnvironmentMode;

        $_SESSION['DB_SERVER'] = $DB_SERVER;
        $_SESSION['DB_USER'] = $DB_USER;
        $_SESSION['DB_PASSWORD'] = $DB_PASSWORD;
        $_SESSION['DB_NAME'] = $DB_NAME;
        $_SESSION['DB_PORT'] = $DB_PORT;

        $_SESSION['CDB_SERVER'] = $CDB_SERVER;
        $_SESSION['CDB_USER'] = $CDB_USER;
        $_SESSION['CDB_PASSWORD'] = $CDB_PASSWORD;
        $_SESSION['CDB_NAME'] = $CDB_NAME;
        $_SESSION['CDB_PORT'] = $CDB_PORT;
    }

    return true;
}
```

```
/*
=== loadSystemConfig() ===
Reads secure/config.txt and loads application directory settings.
*/
function loadSystemConfig($file=null)
{
   global $cftTempUpload,$cftTempDownload,$cftBackups;
   global $cftManual,$cftManualImages;
   global $cftDocFileUploads,$cftDocImageUploads;
   global $FilePhysicalBase;

   $fileRoot = rtrim((string)$FilePhysicalBase, '/\\');
   if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadSystemConfig()."); }

   $defaultDir = $fileRoot . '/secure';

   if ($file === null || trim((string)$file) === '') {
      $file = $defaultDir . '/config.txt';
   } else {
      if (strpos($file,'/') === false && strpos($file,'\\') === false) {
         $file = $defaultDir . '/' . $file;
      }
```

```
}

if (!file_exists($file)) { fatalConfigError("Missing system config file: " . $file); }

$lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
if ($lines === false) { fatalConfigError("Unable to read system config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln,'=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));
    $k = strtolower($k);
    $v = rtrim($v,'/') . '/';

    if ($k === 'tempupload')     { $cftTempUpload = $v; }
    if ($k === 'tempdownload')   { $cftTempDownload = $v; }
    if ($k === 'backups')        { $cftBackups = $v; }
    if ($k === 'manual')         { $cftManual = $v; }
    if ($k === 'manualimages')   { $cftManualImages = $v; }
```

```
        if ($k === 'docfileuploads')  { $cftDocFileUploads = $v; }
        if ($k === 'docimageuploads') { $cftDocImageUploads = $v; }
    }

    if (isset($_SESSION)) {
        $_SESSION['cftTempUpload'] = $cftTempUpload;
        $_SESSION['cftTempDownload'] = $cftTempDownload;
        $_SESSION['cftBackups'] = $cftBackups;
        $_SESSION['cftManual'] = $cftManual;
        $_SESSION['cftManualImages'] = $cftManualImages;
        $_SESSION['cftDocFileUploads'] = $cftDocFileUploads;
        $_SESSION['cftDocImageUploads'] = $cftDocImageUploads;
    }

    return true;
}

/*
=== loadEmailConfig() ===
Reads secure/emailconfig.txt and loads sender, reply-to, admin, support,
and SMTP settings.
*/
```

```
function loadEmailConfig($file=null)
{
   global $mailer,$fromName,$fromEmail,$replyToEmail;
   global $authorEmail,$salesEmail,$supportEmail,$infoEmail,$testEmail,$bounceEmail,$adminEmail;
   global $smtpHost,$smtpPort,$smtpUser,$smtpPassword,$smtpSecure;
   global $FilePhysicalBase;

   $fileRoot = rtrim((string)$FilePhysicalBase, '/\\');
   if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadEmailConfig()."); }

   $defaultDir = $fileRoot . '/secure';

   if ($file === null || trim((string)$file) === '') {
      $file = $defaultDir . '/emailconfig.txt';
   } else {
      if (strpos($file,'/') === false && strpos($file,'\\') === false) {
         $file = $defaultDir . '/' . $file;
      }
   }
```

```
if (!file_exists($file)) { fatalConfigError("Missing email config file: " . $file); }

$lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
if ($lines === false) { fatalConfigError("Unable to read email config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln,'=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));
    $k = strtolower($k);

    if ($k === 'mailer') { $mailer = $v; }
    if ($k === 'fromname') { $fromName = $v; }
    if ($k === 'fromemail') { $fromEmail = $v; }
    if ($k === 'replyto') { $replyToEmail = $v; }

    if ($k === 'authoremail') { $authorEmail = $v; }
    if ($k === 'salesemail') { $salesEmail = $v; }
    if ($k === 'supportemail') { $supportEmail = $v; }
```

```
    if ($k === 'infoemail') { $infoEmail = $v; }
    if ($k === 'testemail') { $testEmail = $v; }
    if ($k === 'bounceemail') { $bounceEmail = $v; }
    if ($k === 'adminemail') { $adminEmail = $v; }

    if ($k === 'smtp_host') { $smtpHost = $v; }
    if ($k === 'smtp_port') { $smtpPort = $v; }
    if ($k === 'smtp_user') { $smtpUser = $v; }
    if ($k === 'smtp_password') { $smtpPassword = $v; }
    if ($k === 'smtp_secure') { $smtpSecure = $v; }
}

if (trim((string)$mailer) === '') { $mailer = 'phpmail'; }
if (trim((string)$smtpPort) === '') { $smtpPort = '587'; }
if (trim((string)$smtpSecure) === '') { $smtpSecure = 'tls'; }

if (isset($_SESSION)) {
    $_SESSION['mailer'] = $mailer;
    $_SESSION['fromName'] = $fromName;
    $_SESSION['fromEmail'] = $fromEmail;
    $_SESSION['replyToEmail'] = $replyToEmail;
```

```
    $_SESSION['authorEmail'] = $authorEmail;
    $_SESSION['salesEmail'] = $salesEmail;
    $_SESSION['supportEmail'] = $supportEmail;
    $_SESSION['infoEmail'] = $infoEmail;
    $_SESSION['testEmail'] = $testEmail;
    $_SESSION['bounceEmail'] = $bounceEmail;
    $_SESSION['adminEmail'] = $adminEmail;

    $_SESSION['smtpHost'] = $smtpHost;
    $_SESSION['smtpPort'] = $smtpPort;
    $_SESSION['smtpUser'] = $smtpUser;
    $_SESSION['smtpPassword'] = $smtpPassword;
    $_SESSION['smtpSecure'] = $smtpSecure;
  }

  return true;
}

/*
=== loadAIConfig() ===
Reads secure/aihost.txt and loads the AI provider, host, API key,
chat model, and embedding model.
```

```
*/
function loadAIConfig($file=null)
{
   global $cftAIProvider,$cftAIHost,$cftAIApiKey,$cftAIChatModel,$cftAIEmbedModel;
   global $FilePhysicalBase;

   $fileRoot = rtrim((string)$FilePhysicalBase, '/\\');
   if ($fileRoot === '') { fatalConfigError("FilePhysicalBase is blank before loadAIConfig()."); }

   $defaultDir = $fileRoot . '/secure';

   if ($file === null || trim((string)$file) === '') {
      $file = $defaultDir . '/aihost.txt';
   } else {
      if (strpos($file,'/') === false && strpos($file,'\\') === false) {
         $file = $defaultDir . '/' . $file;
      }
   }

   if (!file_exists($file)) { fatalConfigError("Missing AI config file: " . $file); }

   $lines = file($file, FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);
```

```
if ($lines === false) { fatalConfigError("Unable to read AI config file: " . $file); }

foreach ($lines as $ln) {
    $ln = trim($ln);
    if ($ln === '') { continue; }
    if ($ln[0] == '#' || $ln[0] == ';') { continue; }
    if (strpos($ln,'=') === false) { continue; }

    list($k,$v) = array_map('trim', explode('=', $ln, 2));
    $k = strtolower($k);

    if ($k === 'provider')   { $cftAIProvider = $v; }
    if ($k === 'host')       { $cftAIHost = $v; }
    if ($k === 'apikey')     { $cftAIApiKey = $v; }
    if ($k === 'chatmodel')  { $cftAIChatModel = $v; }
    if ($k === 'embedmodel') { $cftAIEmbedModel = $v; }
}

if (isset($_SESSION)) {
    $_SESSION['cftAIProvider'] = $cftAIProvider;
    $_SESSION['cftAIHost'] = $cftAIHost;
    $_SESSION['cftAIApiKey'] = $cftAIApiKey;
```

```
    $_SESSION['cftAIChatModel'] = $cftAIChatModel;
    $_SESSION['cftAIEmbedModel'] = $cftAIEmbedModel;
  }

  return true;
}
```

Appendix 4 - Logon and Startup Functions – Code

The following is actual PHP code that controls the User Logon Function and the include start up module that is included in all gui front end programs. It is provided as a sample only however the functionaliy will guide you in converting to the programming language of your choice.

login.php

```
<?php
/*
=== Document Identity ===
DocType: Program
Category: Login.php
Subsystem: Platform Core Authentication and Session Initialization
programName: Login.php
Audience: Both
Overview: Authenticates users, initializes session controls, loads runtime configuration, and redirects successful logins to the main menu.
Keywords: login, logon, authentication, session, security
```

```
=== End Document Identity ===
*/

/*
=== Login Program Purpose ===
This program is the public login screen for the application.
It starts the session, loads shared routines, checks existing login state,
validates submitted credentials, and displays the login form when needed.
*/

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

/* Configure session storage and lifetime before session_start(). */
ini_set('session.save_path', __DIR__ . '/sessionfiles');
ini_set('session.gc_maxlifetime', 28800);
ini_set('session.cookie_lifetime', 28800);

/* Set browser cookie rules for the PHP session. */
session_set_cookie_params([
    'lifetime' => 28800,
```

```
    'path' => '/',
    'secure' => false,
    'httponly' => true,
    'samesite' => 'Lax'
]);

session_start();

/* Create a browser message tracking cookie if one does not already exist. */
if (!isset($_COOKIE['messageUID'])) {
    $messageUID = uniqid('msg_', true);

    setcookie('messageUID', $messageUID, [
        'expires' => time() + 86400,
        'path' => '/',
        'secure' => false,
        'httponly' => true,
        'samesite' => 'Lax'
    ]);

    $_COOKIE['messageUID'] = $messageUID;
}
```

```
/* Load common variables, database routines, and shared utility routines. */
require_once(getcwd()."/include/A_DBUserVariables.php") ;
require_once(getcwd()."/include/A_DBUserRoutines.php") ;
require_once(getcwd()."/include/A_CommonRoutines.php") ;

/* If already logged in, bypass the form and go to the main menu. */
if (!empty($_SESSION['loggedonuserid'])) {
   header("Location: Menu_Main.php");
   exit;
}

/* Store session identity information for later program use. */
$sessionID = session_id();
$_SESSION['sessionID'] = $sessionID;
$sessionIP = $_SERVER['REMOTE_ADDR'];
$_SESSION['sessionIP'] = $sessionIP;

/* Initialize CSRF token for form/session security. */
if (!isset($_SESSION['csrfToken']) || $_SESSION['csrfToken'] == '') {
   $_SESSION['csrfToken'] = bin2hex(random_bytes(32));
}
```

```
/* Initialize login failure tracking and temporary lockout controls. */
if (!isset($_SESSION['loginAttemptCount'])) {
   $_SESSION['loginAttemptCount'] = 0;
}

if (!isset($_SESSION['loginLockUntil'])) {
   $_SESSION['loginLockUntil'] = 0;
}

/* Runtime configuration globals populated by loadRuntimeConfig(). */
global $FilePhysicalBase;
global $FileUrlBase;
global $EnvironmentMode, $DB_SERVER, $DB_USER, $DB_PASSWORD,$DB_NAME,
$DB_PORT;
global $CDB_SERVER,$CDB_USER,$CDB_PASSWORD, $CDB_NAME,$CDB_PORT;
global $cfgTempUpload,$cfgTempDownload,$cfgBackups,$cfgManual,$cfgManualImages;
global $cfgAIProvider, $cfgAIHost,$cfgAIApiKey,$cfgAIChatModel,$cfgAIEmbedModel;

/* Load database, filesystem, URL, and AI runtime settings. */
loadRuntimeConfig();

/* Secondary login check using the alternate session user variable. */
```

```
if (!isset($_SESSION['LoggedOnUserID'])) { $_SESSION['LoggedOnUserID'] = ''; }

if (trim((string)$_SESSION['LoggedOnUserID']) !== '') {
   addMessage("User already logged on.");
   header("Location: Menu_Main.php");
   exit;
}

/* Initialize login processing variables. */
global $userid, $password, $lcUser, $errsw;

$userid = '';
$password = '';
$lcUser = '';
$errsw = 0;

/* Process submitted login form. */
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['btnLogon'])) {
   if (!isset($_SESSION['errmsg'])) { $_SESSION['errmsg'] = ''; }

   $currentTime = time();
```

```
  if ($_SESSION['loginLockUntil'] > $currentTime) {
    $remainingSeconds = $_SESSION['loginLockUntil'] - $currentTime;
    $remainingMinutes = ceil($remainingSeconds / 60);
    addMessage("Too many failed login attempts. Try again in " . $remainingMinutes . "
minutes.");
  } else {
    if ($_SESSION['loginLockUntil'] > 0 && $_SESSION['loginLockUntil'] <= $currentTime) {
      $_SESSION['loginAttemptCount'] = 0;
      $_SESSION['loginLockUntil'] = 0;
    }

    if (isset($_POST['pctUserID'])) { $userid = trim((string)$_POST['pctUserID']); }
    if (isset($_POST['pctPassword'])) { $password = trim((string)$_POST['pctPassword']); }

    if ($userid === '') { addMessage("UserID is required."); }
    if ($password === '') { addMessage("Password is required."); }

    $lcUser = strtolower($userid);

    if (empty($_SESSION['errmsg'])) {
      if (loginUser($userid, $password)) {
        $_SESSION['loginAttemptCount'] = 0;
        $_SESSION['loginLockUntil'] = 0;
```

```
            $lcUser = strtolower($userid);
            $_SESSION['loggedonuserid'] = $lcUser;
            $_SESSION['LoggedOnUserID'] = $lcUser;

            $sec40 = '';
            if (isset($_SESSION['secgroups_array'][40])) { $sec40 =
(string)$_SESSION['secgroups_array'][40]; }

            addMessage("Welcome ". $lcUser . " You have successfully logged in as " .
$_SESSION['UserType'] . ".");

            header("Location: Menu_Main.php");
            exit;
          } else {
            $_SESSION['loginAttemptCount']++;

            if ($_SESSION['loginAttemptCount'] >= 5) {
              $_SESSION['loginLockUntil'] = time() + 600;
              $_SESSION['loginAttemptCount'] = 0;
              addMessage("Too many failed login attempts. Wait 10 minutes before trying
again.");
            } else {
```

```
            addMessage("UserID and Password do not match - Try Again.");
          }
        }
      }
    }
}
?>
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>Production Retrieval-Augmented AI Systems | Ivan Rodriguez</title>
  <meta name="description" content="Engineering AI Knowledge Systems: From Documentation to Intelligent Assistants. A production-ready blueprint for ingesting organizational content into hybrid search and retrieval pipelines.">
  <meta name="keywords" content="<?php echo htmlspecialchars($metaKeywords, ENT_QUOTES, "UTF-8"); ?>">
  <meta name="robots" content="index, follow">

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
```

```
<style>
body{
  margin:0;
  font-family: Arial, Helvetica, sans-serif;
  background: linear-gradient(180deg, #e3f2fd 0%, #f7fbfe 55%, #ffffff 100%);
  color:#0f1720;
}

.wrapLogin{ max-width: 980px; margin: 0 auto; padding: 18px 14px 40px 14px; }

.topbarLogin{
  width:100%;
  background: rgba(255,255,255,0.78);
  border: 1px solid rgba(37,150,190,0.22);
  border-radius: 14px;
  box-shadow: 0 14px 28px rgba(0,0,0,0.06);
  padding: 18px 16px;
  margin-bottom: 18px;
}
.brandCenter{
  display:flex;
```

```
  align-items:center;
  justify-content:center;
  gap:14px;
  flex-wrap:wrap;
  text-align:center;
}
.brandLogo{
  width: 96px;
  height: 96px;
  object-fit: contain;
}
.brandText{ line-height:1.10; }
.brandTitle{
  font-weight: 900;
  letter-spacing: 0.4px;
  font-size: 34px;
  color:#2596be;
  margin:0;
}
.brandSub{
  margin-top:6px;
  font-size: 15px;
```

```
  color:#26323e;
}

.loginCard{
  max-width: 520px;
  margin: 0 auto;
  border: 2px solid #2596be;
  background-color: #e3f2fd;
  border-radius: 14px;
  box-shadow: 0 12px 24px rgba(0,0,0,0.06);
  padding: 16px 16px 14px 16px;
}
.welcomeBox{
  background: rgba(255,255,255,0.80);
  border: 1px solid rgba(37,150,190,0.30);
  border-radius: 12px;
  padding: 10px 12px;
  margin-bottom: 12px;
  color:#0f1720;
  font-size: 14px;
}
```

```
  .loginLabel{ font-weight:700; color:#0f1720; margin-bottom:4px; }
  .loginInput{ border-radius: 10px; border: 1px solid rgba(37,150,190,0.45); }
  .loginRow{ margin-bottom: 10px; }

  .btnRow{ display:flex; justify-content:center; margin-top: 8px; }
</style>
  </head>

<body>

<div class="wrapLogin">

  <div class="topbarLogin">
    <div class="brandCenter">
      <img class="brandLogo" src="images/BookBrandingLogo.png" alt="Symphora">
      <div class="brandText">
        <div class="brandTitle">Production Retrieval-Augmented AI Systems</div>
        <div class="brandSub">AI Knowledge Demo System</div>
      </div>
    </div>
  </div>
```

```
  <center><input type='hidden' maxlength='1' size='1' name='updsw' readonly='readonly' value='<?php echo $updsw; ?>'>

      <?php if (isset($_SESSION['errmsg'])) {echo '<span style="color: red;">' . $_SESSION['errmsg'] . '</span>';unset($_SESSION['errmsg']); } ?>

   </center>


 <div class="loginCard">


   <div class="welcomeBox">

    Welcome. You may login as userid <b>guest</b> password <b>guest</b> to view our system.

   </div>

   <form method="post" action="">

   <div class="loginRow">

    <div class="loginLabel">UserID</div>

    <input type="text" name="pctUserID" id="pctUserID" class="form-control loginInput" value="">

   </div>


   <div class="loginRow">

    <div class="loginLabel">Password</div>

    <input type="password" name="pctPassword" id="pctPassword" class="form-control loginInput" value="">

   </div>
```

```
   <div class="btnRow">
    <button type="submit" name="btnLogon" id="btnLogon" class="btn btn-primary btn-sm">Logon to Demo Website</button>
   </div>
   </form>
  </div>

</div>

</body>
</html>
```

startup.php

<?php

/*

=== Document Identity ===

DocType: Program

Category: startup.php

Subsystem: Platform Core Session Start up All Programs included in all front end programs

programName: startup.php

Audience: Both

Overview: Login to the system, set session variables, and redirect to main menu if already logged in. Also initializes session configuration and includes necessary database and utility files.

Keywords: login, logon, connect

=== End Document Identity ===

*/

/*

=== Program Purpose ===

startup.php is the common startup entry point for the front end application.

This program prepares the PHP runtime session, loads the standard database and utility include files, loads the runtime environment configuration, and establishes the base conditions required before the user is allowed to continue into the

application.

Because this program is included at the beginning of the front end flow, it acts as a platform-level control point. Any program that depends on a valid session, database configuration, shared routines, or login state depends on this startup process being completed first.

=== End Program Purpose ===

*/

/*

=== Session Initialization ===

Only start a session when PHP has not already started one.

The custom session_save_path points PHP session storage to the application-level sessionfiles directory. This keeps session files under the application directory structure instead of relying on the server default session location.

The dirname(__DIR__) expression moves one directory above the current include directory so the sessionfiles directory can be shared consistently across the application.

*/

if (session_status() === PHP_SESSION_NONE) {

```
    session_save_path(dirname(__DIR__) . "/sessionfiles");
    session_start();
}

/*
=== Output Buffering ===
Start output buffering before any page output is produced.

This allows the startup process and downstream programs to safely perform redirects,
set headers, or build page content before anything is sent to the browser.
*/
ob_start();

/*
=== Core Includes ===
Load the standard platform include files required by the application.
```

A_DBUserVariables.php:

Defines common database and user-level variables used by the system.

A_DBUserRoutines.php:

Provides database access routines and user/security related database helpers.

A_CommonRoutines.php:

Provides shared application routines used across front end programs.

These includes are loaded from the current working directory so front end programs can use a consistent include path relative to the executing program.

*/

require_once(getcwd()."/include/A_DBUserVariables.php");

require_once(getcwd()."/include/A_DBUserRoutines.php");

require_once(getcwd()."/include/A_CommonRoutines.php");

/*

=== Runtime Configuration Load ===

Load the runtime configuration after the core include files are available.

This step initializes the environment-specific settings such as database connection values, file paths, URL paths, and other deployment configuration values required by the application.

Keeping this logic inside loadRuntimeConfig() allows the same source code to run in different hosting environments without hard-coding server-specific values inside the front end programs.

```
*/

loadRuntimeConfig();


?>
```

Appendix 5 - API Instruction JSON Schemas – Code

This appendix provides a simplified instructional schema. It is not a production instruction profile. Each organization must design, test, tune, and protect its own API instruction profiles.

The purpose of this appendix is to show the structure of an API instruction contract without exposing proprietary processing logic.

```
{
 "requestTemplate": {
  "requestInfo": {
   "requestId": "...",
   "schemaVersion": "1.0",
   "instructionProfile": "SAMPLE_DOCUMENT_PROCESSING",
   "instructionVersion": "1.0",
   "requestTimestamp": "...",
   "workerType": "PARSE",
   "operationType": "DOCUMENT_PROCESSING"
  },
  "fileInfo": {
   "documentFormat": "PLAIN_TEXT",
   "mimeType": "text/plain",
   "fileSizeBytes": 0,
   "contentTransportFormat": "EXTRACTED_TEXT",
   "fileContentText": ""
```

```
  },
  "documentClassification": {
   "businessDomainHint": "",
   "documentTypeHint": ""
  },
  "structureAndOutput": {
   "responseMode": "FULL_JSON",
   "returnProcessingInfo": "Y",
   "returnUsageInfo": "Y",
   "returnContentInfo": "Y"
  },
  "metadataGuidance": {
   "metadataProfile": "SAMPLE_DOCUMENT_PROCESSING",
   "metadataHint": "The supplied text is already extracted and normalized by the caller.
Use the supplied text as the authoritative input."
  }
 },
 "promptInstructionBlocks": {
  "requestInfoInstructions": {
   "section": "requestInfo",
   "purpose": "Define the request identity, instruction profile, worker type, operation type,
and response contract for this run.",
   "rules": [
    "You must use requestInfo as the control contract for the run.",
    "You must return valid JSON only.",
    "You must not return markdown.",
    "You must not return explanatory prose outside the JSON response.",
    "You must return exactly one processingInfo object.",
    "You must return exactly one usageInfo object.",
    "You must return only the top-level objects defined by the response schema."
   ]
```

```
  },
  "fileInfoInstructions": {
    "section": "fileInfo",
    "purpose": "Define the source text that must be processed.",
    "rules": [
      "You must use fileInfo.fileContentText as the authoritative source text when contentTransportFormat is EXTRACTED_TEXT.",
      "You must not retrieve or infer missing file content.",
      "You must not rewrite, summarize, or expand the supplied source text unless the active instruction profile explicitly requires it.",
      "If fileContentText is blank or unusable, you must return processingInfo.status as ERROR."
    ]
  },
  "documentClassificationInstructions": {
    "section": "documentClassification",
    "purpose": "Provide optional classification hints for processing.",
    "rules": [
      "You may use businessDomainHint as optional context when it is supplied.",
      "You must not invent a business domain if none is supplied.",
      "You must not allow businessDomainHint to override the supplied source text."
    ]
  },
  "processingInfoInstructions": {
    "section": "processingInfo",
    "purpose": "Return processing status, stop reason, warning text, and error text.",
    "rules": [
      "processingInfo.status must be SUCCESS, PARTIAL, or ERROR.",
      "processingInfo.stopReasonCode must contain a short machine-readable value.",
      "processingInfo.stopReasonText must contain a short human-readable explanation.",
      "processingInfo.hasWarningsSw must be Y or N.",
```

```
      "processingInfo.hasErrorsSw must be Y or N.",
      "processingInfo.warningText must contain concise warning text when warnings exist.",
      "processingInfo.errorText must contain concise error text when errors exist.",
      "If processing completed fully, status must be SUCCESS.",
      "If processing completed only part of the supplied input, status must be PARTIAL.",
      "If processing failed or the input is unusable, status must be ERROR.",
      "You must not report SUCCESS when required content was omitted or only partially
processed."
     ]
    },
    "usageInfoInstructions": {
     "section": "usageInfo",
     "purpose": "Return token and usage values known from the run.",
     "rules": [
      "usageInfo.inputTokens must be numeric.",
      "usageInfo.outputTokens must be numeric.",
      "usageInfo.totalTokens must be numeric.",
      "If exact token counts are unavailable, return 0.",
      "You must not invent token counts.",
      "You must not invent cost values.",
      "The caller system is responsible for calculating cost from stored pricing rules."
     ]
    },
    "contentInfoInstructions": {
     "section": "contentInfo",
     "purpose": "Return limited sample content result information.",
     "rules": [
      "You must return contentInfo only in the format required by the response schema.",
      "You must not include unsupported fields.",
      "You must not include markdown.",
      "You must not include explanatory prose outside the JSON response."
```

```
    ]
   }
  },
  "responseSchema": {
   "processingInfo": {
    "status": "",
    "stopReasonCode": "",
    "stopReasonText": "",
    "hasWarningsSw": "N",
    "hasErrorsSw": "N",
    "warningText": "",
    "errorText": "",
    "processingNotes": ""
   },
   "usageInfo": {
    "provider": "",
    "model": "",
    "inputTokens": 0,
    "outputTokens": 0,
    "totalTokens": 0
   },
   "contentInfo": {
    "contentStatus": "",
    "contentItemCount": 0,
    "contentSummary": ""
   }
  }
}
```

Appendix 6 – Standard CRUD and Move Examples – Code

This describes the standard for Crud and Move functions the calls to the database audit write and the database connectivity.

/*

=== Document Identity ===

DocType: Program Example

Category: Standard Database CRUD and Audit Pattern

Subsystem: Database Access and Audit Logging

programName: Standard CRUD Database Access Example

Audience: Both

Overview: Demonstrates database connection selection, generic create/read/update/delete calls, automatic audit logging, and table-specific CRUD wrapper routines.

Keywords: crud, database, connection, audit, insert, select, update, delete

=== End Document Identity ===

*/

/*

=== Standard CRUD Architecture ===

This example has two layers.

The first layer is the generic database access layer. It opens the proper database

connection, builds the SQL statement, executes the statement, and closes the connection.

The second layer is the table-specific wrapper layer. It maps the current pct globals into field arrays and calls the generic routines.

Database audit records are written in the generic createRecord(), updateRecord(), and deleteRecord() routines. That design keeps audit logging centralized so each table-specific CRUD routine does not need its own audit code.

=== End Standard CRUD Architecture ===

*/

/*

=== ConnectToDB() ===

Opens the default operational database connection.

Legacy mode uses the standard four-argument mysqli connection.

Non-legacy mode uses the port-aware five-argument mysqli connection.

*/

```
function ConnectToDB() {
    global $EnvironmentMode, $DB_PORT;
    global $DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME;
```

```
    global $conn;

    if ($EnvironmentMode === 'legacy') {
        $conn = new mysqli($DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME);

        if ($conn->connect_error) {
            die("Legacy DB connection failed: " . $conn->connect_error);
        }

        return $conn;
    }

    $portToUse = (!empty($DB_PORT)) ? (int)$DB_PORT : 3306;

    $conn = new mysqli($DB_SERVER, $DB_USER, $DB_PASSWORD, $DB_NAME,
$portToUse);

    if ($conn->connect_error) {
        die("Cluster DB connection failed: " . $conn->connect_error .
            " (Host=$DB_SERVER, Port=$portToUse)");
    }

    return $conn;
```

```
}

/*
=== ConnectToCDB() ===
Opens the company database connection.

The generic CRUD routines call this function when connectionType is "company".
*/
function ConnectToCDB() {
  global $EnvironmentMode, $DB_PORT;
  global $CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME;
  global $conn;

  if ($EnvironmentMode === 'legacy') {
    $conn = new mysqli($CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME);

    if ($conn->connect_error) {
      die("Legacy DB connection failed: " . $conn->connect_error);
    }

    return $conn;
  }
```

```
    $portToUse = (!empty($DB_PORT)) ? (int)$DB_PORT : 3306;

    $conn = new mysqli($CDB_SERVER, $CDB_USER, $CDB_PASSWORD, $CDB_NAME,
$portToUse);

    if ($conn->connect_error) {
        die("Cluster DB connection failed: " . $conn->connect_error .
            " (Host=$CDB_SERVER, Port=$portToUse)");
    }

    return $conn;
}

/*
=== createRecord() ===
Generic INSERT routine.

This routine selects the database connection, builds the INSERT SQL from the
field array, executes the insert, then writes the database audit record.

AUDIT POINT:
writeDatabaseAudit() is called here after the INSERT executes.
```

```
*/
function createRecord($table, $fields, $connectionType = "default") {
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

    $columns = array_keys($fields);
    $values = [];

    foreach ($fields as $value) {
        $escapedValue = $conn->real_escape_string((string)($value ?? ''));
        $values[] = "'$escapedValue'";
    }

    $sql = "INSERT INTO $table (" . implode(", ", $columns) . ") VALUES (" . implode(", ", $values) . ")";

    $result = $conn->query($sql);

    /*
    === Database Audit Write ===
    Stores the insert event, target table, field values, connection type,
    and success/error result.
    */
    writeDatabaseAudit([
```

```
        "routineName" => "createRecord",
        "tableName" => $table,
        "fields" => $fields,
        "connectionType" => $connectionType,
        "resultStatus" => $result ? "SUCCESS" : "ERROR"
    ]);

    $conn->close();
    return $result;
}

/*
=== readRecord() ===
Generic SELECT routine.

This routine selects the database connection, builds a WHERE clause from the
primary key array, executes the SELECT, returns one row, and closes the
connection.

Read-only access does not write a database audit record.
*/
function readRecord($table, $primaryKeys, $connectionType = "default") {
```

```
    $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

    $conditions = [];

    foreach ($primaryKeys as $key => $value) {
        $conditions[] = "$key = '$value'";
    }

    $sql = "SELECT * FROM $table WHERE " . implode(" AND ", $conditions);

    $result = $conn->query($sql);
    $data = $result ? $result->fetch_assoc() : null;

    $conn->close();
    return $data;
}

/*
=== updateRecord() ===
Generic UPDATE routine.

This routine selects the database connection, builds the SET clause from the
```

```
field array, builds the WHERE clause from the key array, executes the update,
then writes the database audit record.

AUDIT POINT:
writeDatabaseAudit() is called here after the UPDATE executes.
*/
function updateRecord($table, $fields, $primaryKeys, $connectionType = "default") {
  $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

  $updates = [];
  $conditions = [];

  foreach ($fields as $key => $value) {
    $escapedValue = $conn->real_escape_string((string)($value ?? ''));
    $updates[] = "$key = '$escapedValue'";
  }

  foreach ($primaryKeys as $key => $value) {
    $escapedKeyVal = $conn->real_escape_string((string)($value ?? ''));
    $conditions[] = "$key = '$escapedKeyVal'";
  }
```

```
    $sql = "UPDATE $table SET " . implode(", ", $updates) . " WHERE " . implode(" AND ", $conditions);

    $result = $conn->query($sql);

    /*
    === Database Audit Write ===
    Stores the update event, target table, changed field values,
    primary keys, connection type, and success/error result.
    */
    writeDatabaseAudit([
        "routineName" => "updateRecord",
        "tableName" => $table,
        "fields" => $fields,
        "primaryKeys" => $primaryKeys,
        "connectionType" => $connectionType,
        "resultStatus" => $result ? "SUCCESS" : "ERROR"
    ]);

    $conn->close();
    return $result;
}
```

```
/*
=== deleteRecord() ===
Generic DELETE routine.

This routine selects the database connection, builds the WHERE clause from the
primary key array, executes the delete, then writes the database audit record.

AUDIT POINT:
writeDatabaseAudit() is called here after the DELETE executes.
*/
function deleteRecord($table, $primaryKeys, $connectionType = "default") {
  $conn = ($connectionType === "company") ? ConnectToCDB() : ConnectToDB();

  $conditions = [];

  foreach ($primaryKeys as $key => $value) {
    $escapedValue = $conn->real_escape_string((string)($value ?? ''));
    $conditions[] = "$key = '$escapedValue'";
  }

  $sql = "DELETE FROM $table WHERE " . implode(" AND ", $conditions);
```

```
    $result = $conn->query($sql);

    /*
    === Database Audit Write ===
    Stores the delete event, target table, primary keys, connection type,
    and success/error result.
    */
    writeDatabaseAudit([
        "routineName" => "deleteRecord",
        "tableName" => $table,
        "primaryKeys" => $primaryKeys,
        "connectionType" => $connectionType,
        "resultStatus" => $result ? "SUCCESS" : "ERROR"
    ]);

    $conn->close();
    return $result;
}

/*
=== createManual_Chapter_Table() ===
Table-specific CREATE wrapper.
```

Maps Manual_Chapter_Table pct globals into a field array and calls createRecord().

The audit record is written inside createRecord().

*/

function createManual_Chapter_Table() {

  global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw, $pctEditor, $pctLastDate;

  $fields = [

    'BookName' => $pctBookName,

    'ChapterNumber' => $pctChapterNumber,

    'ChapterName' => $pctChapterName,

    'PlacementSw' => $pctPlacementSw,

    'ReadFromSw' => $pctReadFromSw,

    'Editor' => $pctEditor,

    'LastDate' => $pctLastDate

  ];

  return createRecord('Manual_Chapter_Table', $fields, "company");

}

/*

=== readManual_Chapter_Table() ===

Table-specific READ wrapper.

Calls readRecord() to retrieve one Manual_Chapter_Table row.

*/

function readManual_Chapter_Table($keys) {

  return readRecord('Manual_Chapter_Table', $keys, "company");

}

/*

=== updateManual_Chapter_Table() ===

Table-specific UPDATE wrapper.

Maps pct globals into the update field array and calls updateRecord().

The audit record is written inside updateRecord().

*/

function updateManual_Chapter_Table($keys) {

  global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw, $pctEditor, $pctLastDate;

  $fieldsToUpdate = [

    'BookName' => $pctBookName,

    'ChapterNumber' => $pctChapterNumber,

    'ChapterName' => $pctChapterName,

```
      'PlacementSw' => $pctPlacementSw,
      'ReadFromSw' => $pctReadFromSw,
      'Editor' => $pctEditor,
      'LastDate' => $pctLastDate
   ];

   return updateRecord('Manual_Chapter_Table', $fieldsToUpdate, $keys, "company");
}

/*
=== deleteManual_Chapter_Table() ===
Table-specific DELETE wrapper.

Calls deleteRecord() for one Manual_Chapter_Table row.
The audit record is written inside deleteRecord().
*/
function deleteManual_Chapter_Table($keys) {
   return deleteRecord('Manual_Chapter_Table', $keys, "company");
}

/*
=== moveManual_Chapter_Table() ===
```

Table-specific MOVE routine.

Moves one selected database row into the pct globals used by the screen.

*/

function moveManual_Chapter_Table($data) {

  global $pctBookName, $pctChapterNumber, $pctChapterName, $pctPlacementSw, $pctReadFromSw, $pctEditor, $pctLastDate;

  if (isset($data[0]) && is_array($data[0])) { $data = $data[0]; }

  $pctBookName = $data['BookName'];

  $pctChapterNumber = $data['ChapterNumber'];

  $pctChapterName = $data['ChapterName'];

  $pctPlacementSw = $data['PlacementSw'];

  $pctReadFromSw = $data['ReadFromSw'];

  $pctEditor = $data['Editor'];

  $pctLastDate = $data['LastDate'];

}

Appendix 7 – Standard Security Routines – Code

All systems must have security. Here is one example of possibly thousands of routines that exist. It is simple and functional. Security is controlled by functional group numbers, screens records that designate the functional group the screen function belongs to and the User Record that defines by functional security group with Read Wrtie and No Access. Additionally, Security Lever 1 to 5 1 being the highest security level to control internal granular function security.

```
// --------------------------

// Return screen name (no .php)

// --------------------------

function getCleanScreenName() {

   return pathinfo(basename($_SERVER['PHP_SELF']), PATHINFO_FILENAME);

}



// --------------------------

// CHeck all security error out check screen set user access level

// --------------------------

function initSecurityMinimum($requiredLevel = 'R') {

   global $screenName, $screenTitle;

   global $pctSecurityLevel, $pctReadOK, $pctWriteOK;

   global $pctLoggedOnUserID, $pctLoggedOnFirstName, $pctLoggedOnLastName;

   global $pctSecurityGroupsArray, $pctSecurityGroupsString;

   global $pctCompanyID;
```

```
    // 1. Not logged in? Show error + exit
    if (!isUserLoggedIn()) {
        $_SESSION['errmsg'] = "You are not logged on.";
        echo "<div class='container mt-4'><center><span
style='color:red;'>{$_SESSION['errmsg']}<br><a href='index.php'>Return to
Logon</a></span></center></div>";
        exit;
    }

    // 2. Load user/session values
    loadSecurityGlobalsFromSession();

    //addMessage("DEBUG: BEFORE screenName = [" . $screenName . "]");
//addMessage("DEBUG: strlen(screenName) = " . strlen($screenName));

    // 3. Get screen name if not already set
    if (empty($screenName)) {
        $screenName = getCleanScreenName();
    }
    //addMessage("DEBUG: screenName = [" . $screenName . "]");
//addMessage("DEBUG: strlen(screenName) = " . strlen($screenName));
```

```
    // 4. Get group number from DB
    $group = getScreenGroup($screenName);
    //addMessage("DEBUG: group returned = [" . $group . "]");
    if ($group < 1 || $group > 40) {
        $_SESSION['errmsg'] = "Security Group not configured for screen: $screenName.";
        echo "<div class='container mt-4'><center><span
style='color:red;'>{$_SESSION['errmsg']}<br><a
href='index.php'>Return</a></span></center></div>";
        exit;
    }

    // 5. Get access level from security array
    $access = $pctSecurityGroupsArray[$group] ?? 'N';
    $pctSecurityLevel = $access;

    // 6. Set read/write flags
    $pctReadOK = in_array($access, ['R', 'U']);
    $pctWriteOK = ($access === 'U');

    // 7. Enforce required level
    $levels = ['N' => 0, 'R' => 1, 'W' => 2];
    if ($levels[$access] < $levels[$requiredLevel]) {
```

```
    $_SESSION['errmsg'] = "Access Denied: You do not have permission to access screen <strong>$screenName</strong>.";
    echo "<div class='container mt-4'><center><span style='color:red;'>{$_SESSION['errmsg']}<br><a href='index.php'>Return</a></span></center></div>";
    exit;
  }

  return $access; // return actual level: R or U
}

// --------------------------
// Parse 30-character R/U/N string into array [1 => 'R', 2 => 'N'...]
// --------------------------
function parseSecurityString($string) {
  $result = [];
  for ($i = 0; $i < 40; $i++) {
    $result[$i + 1] = strtoupper($string[$i] ?? 'N');
  }
  return $result;
}

// --------------------------
```

```
// Check if user is logged in
// --------------------------
function isUserLoggedIn() {
   return isset($_SESSION['loggedonuserid']) && !empty($_SESSION['loggedonuserid']);
}


// --------------------------
// Store user session after successful login
// --------------------------
function initializeSecuritySession($data) {
   $_SESSION['loggedonuserid'] = $data['Userid'];
   $_SESSION['firstname'] = $data['FirstName'];
   $_SESSION['lastname'] = $data['LastName'];
   $_SESSION['seclevel'] = $data['SecLevel'];
   $_SESSION['UserType'] = $data['WorkType'];


   $secString = '';
   for ($i = 1; $i <= 40; $i++) {
      $field = 'SecGroup' . $i;
      $secString .= $data[$field] ?? 'N';
   }
```

```
    $_SESSION['secgroups'] = $secString;
    $_SESSION['secgroups_array'] = parseSecurityString($secString);
}

// --------------------------
// Load Security Variables
// --------------------------
function loadSecurityGlobalsFromSession() {
    global $pctLoggedOnUserID, $pctLoggedOnFirstName, $pctLoggedOnLastName;
    global $pctSecurityLevel, $pctSecurityGroupsString, $pctSecurityGroupsArray;

    $pctLoggedOnUserID       = $_SESSION['loggedonuserid']    ?? '';
    $pctLoggedOnFirstName    = $_SESSION['firstname']         ?? '';
    $pctLoggedOnLastName     = $_SESSION['lastname']          ?? '';
    $pctSecurityLevel        = $_SESSION['seclevel']          ?? '';
    $pctSecurityGroupsString = $_SESSION['secgroups']         ?? str_repeat('N', 40);
    $pctSecurityGroupsArray  = $_SESSION['secgroups_array']   ?? array_fill(1, 40, 'N');
}

// --------------------------
// Login and validate credentials verifiy hash
// --------------------------
```

```
function loginUser($userid, $password) {

   $keys = ['Userid' => $userid];
   $data = readRecord('Sec_User_Table', $keys);

   // Validate user exists and password matches hash
   if (!$data || !password_verify($password, $data['PassWord'])) {
      $_SESSION['errmsg'] = "Invalid User ID or Password.";
      return false;
   }

   initializeSecuritySession($data);
   return true;
}

// --------------------------
// Get security group assigned to a screen
// --------------------------
function getScreenGroup($screenName) {

   //addMessage("DEBUG SQL ScreenName used = [" . $screenName . "]");
```

```
    $sql = "SELECT SecGroupNumber FROM Sec_Screen_Table WHERE ScreenName =
'$screenName'";
    $result = getRecordsFullSQL($sql, "company");

    $rowCount = is_array($result) ? count($result) : 0;
    //addMessage("DEBUG SQL result count = " . $rowCount);

    if ($result && isset($result[0]['SecGroupNumber'])) {
        return (int)$result[0]['SecGroupNumber'];
    }

    //addMessage("DEBUG: No matching screen found in Sec_Screen_Table.");
    return 0;
}

// --------------------------
// Check user access level to a screen: R, U, or N
// --------------------------
function checkAccess($screenName) {
    if (!isset($_SESSION['secgroups_array'])) return 'N';

    $group = getScreenGroup($screenName);
    if ($group < 1 || $group > 40) return 'N';
```

```
    return $_SESSION['secgroups_array'][$group] ?? 'N';
}

// --------------------------
// Enforce required access level to screen
// --------------------------
function requireSecurity($screenName, $requiredLevel = 'R') {
    $access = checkAccess($screenName);

    $levels = ['N' => 0, 'R' => 1, 'U' => 2];
    if ($levels[$access] < $levels[$requiredLevel]) {
        $_SESSION['errmsg'] = "Access Denied: You do not have permission to access this screen.";
        echo "<div class='container mt-4'><center><span style='color:red;'>{$_SESSION['errmsg']}</span></center></div>";
        exit;
    }

    return true;
}
// --------------------------
// Check if user has 'W' access to a function number (1–40)
```

```
// --------------------------

function checkFunctionAccessByNumber($functionNumber) {

  if (!isset($_SESSION['secgroups_array'])) {

    $_SESSION['errmsg'] = "Security session not initialized.";

    return false;

  }


  if ($functionNumber < 1 || $functionNumber > 40) {

    $_SESSION['errmsg'] = "Invalid function number.";

    return false;

  }


  $access = $_SESSION['secgroups_array'][$functionNumber] ?? 'N';

  return ($access === 'W');

}


// --------------------------

// Optional: Check function-level access (for buttons/features)

// --------------------------

function checkFunctionAccess($groupNumber, $requiredLevel = 'U') {

  if (!isset($_SESSION['secgroups_array'])) return false;
```

```
    $current = $_SESSION['secgroups_array'][$groupNumber] ?? 'N';
    $levels = ['N' => 0, 'R' => 1, 'U' => 2];

    return $levels[$current] >= $levels[$requiredLevel];
}

// fetch the security of a user for a particular function
function getAccessLevel($groupNumber) {
    if (!isset($_SESSION['secgroups_array'])) {
        return 'N'; // Default to 'N' if the security groups array is not set
    }

    if ($groupNumber < 1 || $groupNumber > 40) {
        return 'N'; // Default to 'N' if the group number is out of valid range
    }

    return $_SESSION['secgroups_array'][$groupNumber] ?? 'N';
}
// --------------------------
// Send grop number and requried N R W optional Sec level require
// returns true or false and sends message to customer
// you can call this from the action button or roitne
```

```
// when you allow R for entry but require W or Sec level For update
// -------------------------


function requireGroupAccess($groupNumber, $requiredLevel, $requiredSecLevelMax = 0)
{
   global $screenName, $secLevel;


   $groupNumber = (int)$groupNumber;


   // Normalize requiredLevel
   $requiredLevel = strtoupper(trim((string)$requiredLevel));
   if ($requiredLevel !== 'N' && $requiredLevel !== 'R' && $requiredLevel !== 'W') {
      $requiredLevel = 'N';
   }


   // Pull current level from session (fast path)
   $haveLevel = 'N';
   if (isset($_SESSION['secgroups_array']) && is_array($_SESSION['secgroups_array'])) {
      if (isset($_SESSION['secgroups_array'][(string)$groupNumber])) {
         $haveLevel =
strtoupper(trim((string)$_SESSION['secgroups_array'][(string)$groupNumber]));
      } elseif (isset($_SESSION['secgroups_array'][$groupNumber])) {
```

```
        $haveLevel =
strtoupper(trim((string)$_SESSION['secgroups_array'][$groupNumber]));
      }
   }

   if ($haveLevel !== 'N' && $haveLevel !== 'R' && $haveLevel !== 'W') {
      $haveLevel = 'N';
   }

   // Compare N/R/W as ordered levels
   $rank = array('N' => 0, 'R' => 1, 'W' => 2);

   $groupOk = ($rank[$haveLevel] >= $rank[$requiredLevel]);

   // Optional SecLevel gate (1 is highest authority, 5 is lowest)
   $secOk = true;
   $requiredSecLevelMax = (int)$requiredSecLevelMax;
   if ($requiredSecLevelMax > 0) {
      $userSecLevel = (int)$secLevel;
      $secOk = ($userSecLevel > 0 && $userSecLevel <= $requiredSecLevelMax);
   }

   if ($groupOk && $secOk) {
```

```
    return true;
  }

  // On failure, try to get group name (company DB)
  $groupName = '';
  $where = "SecGroupNumber=".$groupNumber;
  $groupName = trim((string)getSingleValue("Sec_Group_Table", "SecGroupName", $where, "company"));

  $screen = trim((string)$screenName);
  if ($screen === '') { $screen = 'UnknownScreen'; }

  // Build compact 2-3 line message
  $line1 = "Sorry - not authorized. Screen: ".$screen.".";
  $line2 = "Group ".$groupNumber.($groupName !== '' ? " ".$groupName : "").": Your ".$haveLevel.", Required ".$requiredLevel.".";

  if ($requiredSecLevelMax > 0) {
    $userSecLevel = (int)$secLevel;
    $line3 = "SecLevel: Your ".$userSecLevel.", Required ".$requiredSecLevelMax.". Contact your Security Administrator to update your profile.";
    addCommonMessage($line1." ".$line2." ".$line3);
  } else {
```

```
        $line3 = "Contact your Security Administrator to update your profile for this group.";
        addCommonMessage($line1." ".$line2." ".$line3);
    }

    return false;
}
```

Appendix 8 – Inverted Index Search – Code

This section represents in PHP code formatable to any language the basic Inverted Index Search Module.

```
<?php

/*=== Document Identity ===

DocType: Program

Category: Screens and UI

Subsystem: Inverted Index

ProgramName: Search.php

Audience: Both

Overview: Search screen presented to customer when main menu keyword search invoked

Keywords: Search, Inverted Index, Candidate Retrieval, JSON

=== End Document Identity === */


/**

 * SearchInclude.php

 * ------------------------------------------------------------------------

 * Runtime search engine against the manual/procedure inverted index.

 *

 * Entry point:

 *   array searchExecute(string $query, int $offset, int $limit, array $options = [])

 *    Returns:

 *     [
```

```
*       'rows'   => [          // page-sliced, sorted results ready for UI
*         [
*           'DocID'       => int,
*           'SourceType'    => 'SCR'|'PROC',
*           'SourceName'    => string,
*           'Title'       => string,
*           'BookName'     => string,
*           'ChapterName'   => string,
*           'SectionNumber' => string,
*           'SectionName'   => string,
*           'HelpMode'     => 'Text'|'HTML',
*           'LastIndexed'   => 'YYYY-MM-DD HH:MM:SS',
*           'score'       => float,
*           'matchMask'    => int,     // OR of field masks across matched terms
*           'snippet'      => string    // plain-text (no HTML entities)
*         ],
*         ...
*       ],
*       'total'  => int,       // total documents matched (before paging)
*       'tookMs' => int        // elapsed milliseconds
*     ]
*
```

* Notes:

* - AND semantics across all query tokens (always “search everything”).

* - Field weighting via FieldMask. Current mask assignments:

*

* 1 = title (t)

* 2 = section (s)

* 4 = chapter (c)

* 8 = meta (m)

* 16 = body (b)

* 32 = alt (a)

* 64 = bookName (n)

* 128 = docType (d)

* 256 = metaData (x)

* 512 = keywords (k)

*

* - Tokens may originate from any of the above channels. Query tokens must

* match across the document using AND semantics regardless of field.

*

* - Phrase/proximity boost uses the stored Positions JSON. Positions may

* contain the following field keys:

*

* t,s,c,m,b,a,n,d,x,k

```
*
* - Snippet is built here (joins back to source tables) and is returned as
*   plain text without HTML entities.
*
* - All DB access uses helper functions and the company database connection.
*
* - Query logging to Search_QueryLog_Table using getMaxValue()+1 key generation.
*
* Config (optional, pulled from Search_Config_Table; defaults used if missing):
*
*   search.maxAllLimit          default 500    // hard cap when UI passes “All”
*   search.maxCandidateDocs     default 5000   // union cap safety (rare in AND mode)
*
*   search.weights.title        default 5.0
*   search.weights.section      default 2.0
*   search.weights.chapter      default 1.5
*   search.weights.meta         default 1.2
*   search.weights.body         default 1.0
*   search.weights.alt          default 0.5
*
*   search.weights.bookName     default 0.75
*   search.weights.docType      default 0.50
```

```
*   search.weights.metaData      default 0.90
*   search.weights.keywords      default 1.10
*
*   search.idfFloor              default 0.1
*   search.phraseBoost           default 1.25
*
*   search.snippet.preferHitLines default 2
*   search.snippet.fallbackLines  default 3
*   search.snippet.maxLength     default 240
*/

/* ============================================================
  IMPORTANT TOKENIZATION RULES

  1. Search tokens are normalized to lowercase a-z0-9 only.
     Underscores and punctuation are removed during tokenization.

  2. Identifier fields such as:
       ScreenName
       ProcedureName
       SourceName
     MUST NOT be token-normalized.
```

```
    These values frequently contain underscores (e.g. SN_W_Add)
    and must be preserved exactly for SQL lookups.

  3. Therefore:
    - Token lists use normalized token helpers.
    - Name lists use _srch_sql_quote_list() and preserve '_'.

  ============================================================ */


function searchExecute($query, $offset, $limit, $options = []) {
   // Purpose: Execute a runtime AND search over the inverted index (OR within synonym groups),
   // score matches with field weights + IDF and optional phrase/proximity boost, then return
   // sorted, paged results ready for the Search.php UI.

   $t0 = microtime(true);

   // ---------- Normalize input & tokens ----------
   $qraw   = (string)$query;
   $qnorm  = _srch_norm_text($qraw);
```

```
$tokens = _srch_tokenize($qnorm);

// Stopword filter
$stop   = _srch_load_stopwords(); // set of normalized stopwords
$tokens = array_values(array_filter($tokens, function($tok) use ($stop){
   return $tok !== '' && !isset($stop[$tok]);
}));

if (count($tokens) === 0) {
   $tookMs = (int)round((microtime(true)-$t0)*1000);
   _srch_log_query($qraw, 0, $tookMs);
   return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ---------- Config ----------
$cfg = _srch_load_config([
   'search.maxAllLimit'          => '500',
   'search.maxCandidateDocs'     => '5000',

   'search.weights.title'        => '5.0',
   'search.weights.section'      => '2.0',
   'search.weights.chapter'      => '1.5',
```

```
    'search.weights.meta'         => '1.2',
    'search.weights.body'         => '1.0',
    'search.weights.alt'          => '0.5',

    // New index channels (safe defaults even if not present in Search_Config_Table yet)
    'search.weights.bookName'     => '0.75',
    'search.weights.docType'      => '0.50',
    'search.weights.metaData'     => '0.90',
    'search.weights.keywords'     => '1.10',

    'search.idfFloor'             => '0.1',
    'search.phraseBoost'          => '1.25',
    'search.snippet.preferHitLines' => '2',
    'search.snippet.fallbackLines'  => '3',
    'search.snippet.maxLength'    => '240',
  ]);

  $maxAllLimit     = isset($options['maxAllLimit'])     ? (int)$options['maxAllLimit']     :
(int)$cfg['search.maxAllLimit'];
  $maxCandidateDocs = isset($options['maxCandidateDocs']) ?
(int)$options['maxCandidateDocs'] : (int)$cfg['search.maxCandidateDocs'];

  $weights = [
```

```
    'title'    => (float)$cfg['search.weights.title'],
    'section'  => (float)$cfg['search.weights.section'],
    'chapter'  => (float)$cfg['search.weights.chapter'],
    'meta'     => (float)$cfg['search.weights.meta'],
    'body'     => (float)$cfg['search.weights.body'],
    'alt'      => (float)$cfg['search.weights.alt'],

    'bookName' => (float)$cfg['search.weights.bookName'],
    'docType'  => (float)$cfg['search.weights.docType'],
    'metaData' => (float)$cfg['search.weights.metaData'],
    'keywords' => (float)$cfg['search.weights.keywords'],
];

$idfFloor    = max(0.000001, (float)$cfg['search.idfFloor']);
$phraseBoost = max(1.0, (float)$cfg['search.phraseBoost']);

$preferHitLines = max(0,  (int)$cfg['search.snippet.preferHitLines']);
$fallbackLines  = max(1,  (int)$cfg['search.snippet.fallbackLines']);
$snippetMax     = max(60, (int)$cfg['search.snippet.maxLength']);

// ---------- Synonym expansion -> groups of terms ----------
// OR within a group, AND across groups
```

```
$exp           = _srch_expand_synonyms($tokens);
$queryGroupsTerms = $exp['groups'];
$origSet       = $exp['origSet'];

// Resolve all terms in one SQL -> TokenID
$allTermsSet = [];
foreach ($queryGroupsTerms as $g) {
   foreach ($g as $t) { $allTermsSet[$t] = true; }
}
$allTerms = array_keys($allTermsSet);

if (empty($allTerms)) {
   $tookMs = (int)round((microtime(true)-$t0)*1000);
   _srch_log_query($qraw, 0, $tookMs);
   return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// Tokens are normalized a-z0-9; safe to quote list
$inList  = _srch_sql_quote_list($allTerms);

$rowsTok = getRecordsFullSQL(
   "SELECT TokenID, Token, DocFreq
```

```
    FROM Search_Token_Table
    WHERE Token IN ($inList)",
   'company',
   'search.syn.tokresolve'
);

$tokIdByTerm = [];   // term => TokenID
$dfByTokId   = [];   // TokenID => DocFreq
$termByTokId = [];   // TokenID => term

if (is_array($rowsTok)) {
   foreach ($rowsTok as $r) {
      if (!is_array($r) || !isset($r['Token'])) continue;

      $tid = isset($r['TokenID']) ? (int)$r['TokenID'] : 0;
      if ($tid <= 0) continue;

      $tk  = strtolower((string)$r['Token']);
      $tokIdByTerm[$tk] = $tid;
      $dfByTokId[$tid]  = max(1, isset($r['DocFreq']) ? (int)$r['DocFreq'] : 1);
      $termByTokId[$tid]= $tk;
   }
```

```
}

// Rebuild groups in TokenID space (drop unknown terms)
$queryGroups = [];
foreach ($queryGroupsTerms as $g) {
  $lst = [];
  foreach ($g as $t) {
    if (isset($tokIdByTerm[$t])) $lst[] = $tokIdByTerm[$t];
  }
  if (!empty($lst)) $queryGroups[] = array_values(array_unique($lst));
}

if (empty($queryGroups)) {
  $tookMs = (int)round((microtime(true)-$t0)*1000);
  _srch_log_query($qraw, 0, $tookMs);
  return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ---------- Grouped AND matching (build $docSet, $docAgg) ----------
$idfCache = [];       // tokenId => idf
$docSet   = [];       // DocID => true
$docAgg   = [];       // DocID => ['score'=>float, 'mask'=>int, 'pos'=> [tokenId => positions] ]
```

```
$synFactorExact = 1.0;

$synFactorAlias = 0.85;


$Ndocs = _srch_total_docs();


// Seed: union of first group

$g0 = $queryGroups[0];


foreach ($g0 as $tid) {


  if (!isset($idfCache[$tid])) {

    $df = isset($dfByTokId[$tid]) ? $dfByTokId[$tid] : 1;

    $idfCache[$tid] = _srch_idf($Ndocs, $df, $idfFloor);

  }


  $posts = _srch_load_postings_for_token($tid);


  $term    = isset($termByTokId[$tid]) ? $termByTokId[$tid] : '';

  $isExact = ($term !== '' && isset($origSet[$term]));

  $synW    = $isExact ? $synFactorExact : $synFactorAlias;
```

```
foreach ($posts as $p) {

    $docId = (int)$p['DocID'];
    $tf    = (int)$p['TermFreq'];
    $mask  = (int)$p['FieldMask'];
    $posJ  = $p['Positions'];

    $docSet[$docId] = true;

    $add = $tf * _srch_weight_from_mask($mask, $weights) * $idfCache[$tid] * $synW;

    if (!isset($docAgg[$docId])) {

        $docAgg[$docId] = [
            'score' => $add,
            'mask'  => $mask,
            'pos'   => [ $tid => _srch_positions_decode($posJ) ],
        ];

    } else {

        $docAgg[$docId]['score'] += $add;
```

```
        $docAgg[$docId]['mask']  |= $mask;


        if (!isset($docAgg[$docId]['pos'][$tid])) {
           $docAgg[$docId]['pos'][$tid] = _srch_positions_decode($posJ);
        }
      }
   }
}

// For each remaining group: union within group, then INTERSECT with running set
for ($gi = 1; $gi < count($queryGroups); $gi++) {

   if (empty($docSet)) break;

   $grpSet = [];
   $grpAgg = [];

   $currentDocs = array_keys($docSet);

   foreach ($queryGroups[$gi] as $tid) {

      if (!isset($idfCache[$tid])) {
```

```
    $df = isset($dfByTokId[$tid]) ? $dfByTokId[$tid] : 1;
    $idfCache[$tid] = _srch_idf($Ndocs, $df, $idfFloor);
}

$posts = _srch_load_postings_for_token_limited($tid, $currentDocs);

$term    = isset($termByTokId[$tid]) ? $termByTokId[$tid] : '';
$isExact = ($term !== '' && isset($origSet[$term]));
$synW    = $isExact ? $synFactorExact : $synFactorAlias;

foreach ($posts as $p) {

    $docId = (int)$p['DocID'];
    $tf    = (int)$p['TermFreq'];
    $mask  = (int)$p['FieldMask'];
    $posJ  = $p['Positions'];

    $grpSet[$docId] = true;

    $add = $tf * _srch_weight_from_mask($mask, $weights) * $idfCache[$tid] * $synW;

    if (!isset($grpAgg[$docId])) {
```

```
            $grpAgg[$docId] = [
               'score' => $add,
               'mask'  => $mask,
               'pos'   => [ $tid => _srch_positions_decode($posJ) ],
            ];

         } else {

            if ($add > $grpAgg[$docId]['score']) $grpAgg[$docId]['score'] = $add;
            $grpAgg[$docId]['mask'] |= $mask;

            if (!isset($grpAgg[$docId]['pos'][$tid])) {
               $grpAgg[$docId]['pos'][$tid] = _srch_positions_decode($posJ);
            }
         }
      }
   }

   // AND: intersect with running set
   $newSet = array_intersect_key($docSet, $grpSet);
```

```
if (empty($newSet)) {
   $docSet = [];
   $docAgg = [];
   break;
}

// Merge this group's contributions into aggregate (only for surviving docs)
$kept = [];

foreach ($newSet as $docId => $_) {

   $docId = (int)$docId;

   if (!isset($docAgg[$docId]) || !isset($grpAgg[$docId])) continue;

   $prev = $docAgg[$docId];
   $ga   = $grpAgg[$docId];

   $prev['score'] += $ga['score'];
   $prev['mask']  |= $ga['mask'];

   foreach ($ga['pos'] as $tid => $posArr) {
```

```
            if (!isset($prev['pos'][$tid])) $prev['pos'][$tid] = $posArr;
        }

        $kept[$docId] = $prev;
    }

    $docSet = $newSet;
    $docAgg = $kept;

    // Defensive cap
    if (count($docSet) > $maxCandidateDocs) {

        $docSetIds = array_slice(array_keys($docSet), 0, $maxCandidateDocs);
        $docSet = array_fill_keys($docSetIds, true);
        $docAgg = array_intersect_key($docAgg, $docSet);
    }
}

// No matches?
$total = count($docSet);

if ($total === 0) {
```

```
    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, 0, $tookMs);
    return ['rows'=>[], 'total'=>0, 'tookMs'=>$tookMs];
}

// ---------- Phrase/proximity boost ----------
if ($phraseBoost > 1.0 && !empty($queryGroups)) {

    $tokSet = [];
    foreach ($queryGroups as $g) { foreach ($g as $tid) { $tokSet[$tid] = true; } }
    $tokenIds  = array_keys($tokSet);

    // Keep phrase checks on original text-like fields only
    $fieldKeys = ['t','s','c','m','b','a'];

    foreach ($docAgg as $docId => $agg) {

        $adjFound = false;

        foreach ($fieldKeys as $fk) {

            $perTok = [];
```

```
foreach ($tokenIds as $tid) {
   if (isset($agg['pos'][$tid][$fk]) && is_array($agg['pos'][$tid][$fk])) {
      $perTok[$tid] = $agg['pos'][$tid][$fk];
   }
}

if (count($perTok) >= 2) {

   $tids = array_keys($perTok);

   for ($i = 0; $i < count($tids) && !$adjFound; $i++) {
      for ($j = $i + 1; $j < count($tids) && !$adjFound; $j++) {
         if (_srch_has_adjacent($perTok[$tids[$i]], $perTok[$tids[$j]])) {
            $adjFound = true;
            break;
         }
      }
   }
}

if ($adjFound) break;
```

```
      }

      if ($adjFound) $docAgg[$docId]['score'] *= $phraseBoost;
    }
}

// ---------- Join to doc metadata & sort ----------
$docIds   = array_keys($docAgg);
$docsMeta = _srch_load_docs_meta($docIds);

$items = [];

foreach ($docAgg as $docId => $agg) {

   if (!isset($docsMeta[$docId])) continue;

   $m = $docsMeta[$docId];

   $items[] = [
      'DocID'       => $docId,
      'SourceType'    => $m['SourceType'],
      'SourceName'    => $m['SourceName'],
```

```
      'BookName'     => isset($m['BookName']) ? $m['BookName'] : '',
      'Title'        => $m['Title'],
      'ChapterName'   => $m['ChapterName'],
      'SectionNumber' => $m['SectionNumber'],
      'SectionName'   => $m['SectionName'],
      'HelpMode'     => '',
      'LastIndexed'   => $m['LastIndexed'],
      'score'        => (float)$agg['score'],
      'matchMask'    => (int)$agg['mask'],
    ];
}

usort($items, function($a, $b) {

    if ($a['score'] !== $b['score']) return ($a['score'] > $b['score']) ? -1 : 1;

    if ($a['LastIndexed'] !== $b['LastIndexed']) {
      return strcmp($b['LastIndexed'], $a['LastIndexed']);
    }

    $t = strcmp((string)$a['Title'], (string)$b['Title']);
    if ($t !== 0) return $t;
```

```
    return $a['DocID'] <=> $b['DocID'];
});

// ---------- Paging ----------
$totalHits  = count($items);
$pageLimit  = ($limit > 0) ? (int)$limit : min($totalHits, $maxAllLimit);
$pageOffset = max(0, (int)$offset);

if ($pageOffset >= $totalHits) $pageOffset = 0;

$pageSlice = array_slice($items, $pageOffset, $pageLimit);

// ---------- Load text & build snippets for page slice ----------
$scrNames = [];
$procNames = [];

foreach ($pageSlice as $it) {
    if ($it['SourceType'] === 'SCR')  $scrNames[]  = $it['SourceName'];
    if ($it['SourceType'] === 'PROC') $procNames[] = $it['SourceName'];
}
```

```
  $scrMap  = _srch_load_screen_texts($scrNames);
  $procMap = _srch_load_proc_texts($procNames);

  foreach ($pageSlice as &$it) {

    $nm = $it['SourceName'];

    if ($it['SourceType'] === 'SCR') {

      $hm = isset($scrMap[$nm]) ? $scrMap[$nm]['HelpMode'] : 'Text';
      $tx = isset($scrMap[$nm]) ? $scrMap[$nm]['Text']     : '';

      $it['HelpMode'] = $hm;
      $it['snippet']  = _srch_make_snippet($tx, $tokens, $preferHitLines, $fallbackLines,
$snippetMax);

    } else {

      $hm = isset($procMap[$nm]) ? $procMap[$nm]['HelpMode'] : 'Text';
      $tx = isset($procMap[$nm]) ? $procMap[$nm]['Text']     : '';

      $it['HelpMode'] = $hm;
```

```
        $it['snippet'] = _srch_make_snippet($tx, $tokens, $preferHitLines, $fallbackLines, $snippetMax);
      }
    }

    unset($it);

    // ---------- Log & return ----------
    $tookMs = (int)round((microtime(true)-$t0)*1000);
    _srch_log_query($qraw, (int)$total, $tookMs);

    return ['rows'=>$pageSlice, 'total'=>$total, 'tookMs'=>$tookMs];
}




// ********************
// Helper functions and globals for SearchInclude.php
// ********************
```

```
/* ========================= Helpers ========================= */

function _srch_norm_text($s) {
   // Purpose: Normalize inbound text to a consistent form for search processing.
   // - Decode HTML entities
   // - Lowercase
   // - Replace punctuation with spaces
   // - Collapse whitespace
   //
   // NOTE: Normalize the inbound query string in Search.php before calling searchExecute().
   $s = html_entity_decode($s, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
   $s = mb_strtolower($s, 'UTF-8');
   $s = preg_replace('/[^\p{L}\p{Nd}]+/u', ' ', $s);
   $s = preg_replace('/\s+/u', ' ', $s);
   return trim($s);
}

function _srch_tokenize($s) {
   // Purpose: Convert a normalized query string into a stable token list.
   // - Uses _srch_norm_text() first so token boundaries match normalization rules
   // - Produces a-z0-9 tokens only (safe for SQL and consistent with token tables)
   // - De-duplicates while preserving order
```

```
    $s = _srch_norm_text((string)$s);
    if ($s === '') return [];

    $parts = preg_split('/\s+/u', $s, -1, PREG_SPLIT_NO_EMPTY);

    $tokens = [];
    foreach ($parts as $p) {
        $t = preg_replace('/[^a-z0-9]/', '', strtolower((string)$p));
        if ($t !== '') $tokens[] = $t;
    }

    $seen = [];
    $out  = [];
    foreach ($tokens as $t) {
        if (isset($seen[$t])) continue;
        $seen[$t] = true;
        $out[] = $t;
    }
    return $out;
}

/**
```

```
* Expand query tokens using Search_Synonym_Table.
* Returns:
*   [
*     'groups'  => [ ['tokA','tokB',...], ['tokC'], ... ], // OR within each group; AND across groups
*     'origSet' => [ 'tokA' => true, 'tokC' => true, ... ] // originals for scoring (exact vs synonym)
*   ]
*
* Notes:
* - Expects tokens already normalized the same way you index (lowercase, a-z0-9).
* - getRecordsFullSQL requires connectionType = 'company' here because these tables
*   have been moved to different DBs and this search synonym table is in company DB.
*/
function _srch_expand_synonyms(array $tokens) {
  $out = ['groups' => [], 'origSet' => []];
  if (empty($tokens)) return $out;

  // Track originals so we can slightly down-weight synonyms later
  foreach ($tokens as $t) { $out['origSet'][$t] = true; }

  // Build IN (...) safely using your existing helper
  $inList = _srch_sql_quote_list($tokens);
```

```
// Pull ALL tokens from the synonym groups that contain ANY of the query tokens
$sql = "
  SELECT s2.SynGroupID, s2.Token AS SynTok
  FROM Search_Synonym_Table s1
  JOIN Search_Synonym_Table s2 ON s2.SynGroupID = s1.SynGroupID
  WHERE s1.Token IN ($inList)
";
$rows = getRecordsFullSQL($sql, 'company', 'search.syn.expand');

$groupMap = [];       // gid => set(token => true)
$seenInAnyGroup = [];  // token => true (tokens that appear in any group)

if (is_array($rows)) {
  foreach ($rows as $r) {
    if (!is_array($r)) continue;

    $gid = isset($r['SynGroupID']) ? (int)$r['SynGroupID'] : 0;
    $tok = isset($r['SynTok']) ? strtolower((string)$r['SynTok']) : '';

    // Normalize synonym token to the same a-z0-9 form used everywhere else
    $tok = preg_replace('/[^a-z0-9]/', '', $tok);
    if ($gid <= 0 || $tok === '') continue;
```

```
            if (!isset($groupMap[$gid])) $groupMap[$gid] = [];
            $groupMap[$gid][$tok] = true;
            $seenInAnyGroup[$tok] = true;
        }
    }

    // Add each found synonym group (OR within group)
    foreach ($groupMap as $gid => $set) {
        $out['groups'][] = array_values(array_unique(array_keys($set)));
    }

    // Add singleton groups for tokens not present in any synonym group
    foreach ($tokens as $t) {
        if (!isset($seenInAnyGroup[$t])) {
            $out['groups'][] = [$t];
        }
    }

    return $out;
}
```

```
function _srch_load_stopwords() {
   // Purpose: Load active stopwords and normalize them into the same token
   // format used by query tokenization (lowercase a-z0-9).
   // NOTE: getRecordsFullSQL requires connectionType = 'company' because
   // these search support tables are stored in the company database.

   $set = [];

   $sql = "SELECT Token FROM Search_Stopword_Table WHERE IsActive='Y'";
   $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.stopwords');

   foreach ($rows as $r) {

      if (!is_array($r) || !isset($r['Token'])) continue;

      $tok = strtolower((string)$r['Token']);

      // Normalize to same form as query tokens
      $tok = preg_replace('/[^a-z0-9]/', '', $tok);

      if ($tok === '') continue;
```

```
      $set[$tok] = true;
    }

    return $set;
}

function _srch_load_config($defaults) {
    // Purpose: Load search configuration overrides from Search_Config_Table,
    // falling back to provided defaults.
    // NOTE: getRecordsFullSQL requires connectionType = 'company' because
    // this configuration table resides in the company database.

    $keys = array_keys($defaults);
    $in = _srch_sql_quote_list($keys);

    $sql = "SELECT ConfigKey, ConfigVal FROM Search_Config_Table WHERE ConfigKey IN
($in)";
    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.config');

    $map = $defaults;

    foreach ($rows as $r) {
        if (!is_array($r) || !isset($r['ConfigKey'])) continue;
```

```
    $map[(string)$r['ConfigKey']] = (string)$r['ConfigVal'];
  }

  return $map;
}

function _srch_total_docs() {
  // Purpose: Return total indexed document count (cached) for IDF calculations.
  // NOTE: getSingleValueFromSQL requires connectionType = 'company' because
  // Search_Doc_Table resides in the company database.

  static $N = null;

  if ($N !== null) return $N;

  $N = (int)getSingleValueFromSQL(
    "SELECT COUNT(*) FROM Search_Doc_Table",
    'company',
    'searchExecute.totalDocs'
  );

  return $N;
```

```
}

function _srch_idf($N, $df, $floor) {
   // Purpose: Compute IDF-like score component with floor to avoid extreme values.

   $df = max(1, (int)$df);
   $N  = max($df, (int)$N);

   $idf = log(($N + 1.0) / ($df + $floor)) + 1.0;

   return $idf;
}

function _srch_weight_from_mask($mask, $w) {
   // Purpose: Convert FieldMask bitset into summed field weight for scoring.
   //
   // Mask bits:
   //  1   = title (t)
   //  2   = section (s)
   //  4   = chapter (c)
   //  8   = meta (m)
   //  16  = body (b)
```

```
    //  32  = alt (a)
    //  64  = bookName (n)
    //  128 = docType (d)
    //  256 = metaData (x)
    //  512 = keywords (k)

    $sum = 0.0;

    if (($mask & 1)   !== 0) $sum += $w['title'];
    if (($mask & 2)   !== 0) $sum += $w['section'];
    if (($mask & 4)   !== 0) $sum += $w['chapter'];
    if (($mask & 8)   !== 0) $sum += $w['meta'];
    if (($mask & 16)  !== 0) $sum += $w['body'];
    if (($mask & 32)  !== 0) $sum += $w['alt'];

    if (($mask & 64)  !== 0) $sum += $w['bookName'];
    if (($mask & 128) !== 0) $sum += $w['docType'];
    if (($mask & 256) !== 0) $sum += $w['metaData'];
    if (($mask & 512) !== 0) $sum += $w['keywords'];

    return $sum > 0 ? $sum : $w['body']; // fallback
}
```

```
function _srch_positions_decode($JSON) {
   // Purpose: Decode Positions JSON into stable structure for phrase/proximity checks.
   //
   // Possible keys:
   //  t,s,c,m,b,a,n,d,x,k

   $empty = [
      't'=>[], 's'=>[], 'c'=>[], 'm'=>[], 'b'=>[], 'a'=>[],
      'n'=>[], 'd'=>[], 'x'=>[], 'k'=>[]
   ];

   if ($JSON === null || $JSON === '') {
      return $empty;
   }

   $arr = JSON_decode($JSON, true);

   if (!is_array($arr)) return $empty;

   foreach (['t','s','c','m','b','a','n','d','x','k'] as $k) {
      if (!isset($arr[$k]) || !is_array($arr[$k])) {
```

```
        $arr[$k] = [];
      }
    }

    return $arr;
}

function _srch_has_adjacent($a, $b) {
    // Purpose: Return true if any position in $a is adjacent (+/- 1) to any in $b
    // for a small phrase/proximity boost.

    if (empty($a) || empty($b)) return false;

    // Use a set for faster membership
    $set = array_fill_keys($a, true);

    foreach ($b as $pos) {
      $p = (int)$pos;
      if (isset($set[$p - 1]) || isset($set[$p + 1])) return true;
    }
    return false;
}
```

```
function _srch_sql_quote_list($vals) {
   // Purpose: Build a quoted SQL IN(...) list from token strings.
   // Tokens are expected to be normalized to [a-z0-9]. We enforce that again
   // here for safety and determinism.

   $out = [];

   foreach ($vals as $v) {
      $t = strtolower((string)$v);
      $t = preg_replace('/[^a-z0-9]/', '', $t);
      if ($t === '') continue;
      $out[] = "'" . $t . "'";
   }

   return empty($out) ? "''" : implode(',', $out);
}

function _srch_sql_in_numbers($nums) {
   // Purpose: Build a numeric SQL IN(...) list from an array of values.

   $nums = array_values(array_filter($nums, function($n){ return is_numeric($n); }));
```

```
    if (empty($nums)) return 'NULL';

    return implode(',', array_map('intval', $nums));
}

function _srch_load_postings_for_token($tokenId) {
    // Purpose: Load ALL postings rows for a single TokenID from Search_Posting_Table.
    // NOTE: getRecordsFullSQL requires connectionType = 'company' because these search
    // index tables are stored in the company database.

    $sql = "SELECT TokenID, DocID, TermFreq, FieldMask, Positions
            FROM Search_Posting_Table
            WHERE TokenID = " . (int)$tokenId;

    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.postings1');

    $out = [];

    foreach ($rows as $r) {
        if (!is_array($r)) continue;

        $out[] = [
```

```
            'TokenID'   => isset($r['TokenID'])   ? (int)$r['TokenID']   : (int)$tokenId,
            'DocID'     => isset($r['DocID'])     ? (int)$r['DocID']     : 0,
            'TermFreq'  => isset($r['TermFreq'])  ? (int)$r['TermFreq']  : 0,
            'FieldMask' => isset($r['FieldMask']) ? (int)$r['FieldMask'] : 0,
            'Positions' => isset($r['Positions']) ? (string)$r['Positions'] : '',
        ];
    }

    return $out;
}

function _srch_load_postings_for_token_limited($tokenId, $docIds) {
    // Purpose: Load postings for a TokenID limited to a known set of DocIDs
    // (used after candidate set is narrowed).
    // NOTE: getRecordsFullSQL requires connectionType = 'company' because these search
    // index tables are stored in the company database.

    if (empty($docIds)) return [];

    $out = [];
    $chunkSize = 900;
```

```
for ($i = 0; $i < count($docIds); $i += $chunkSize) {

   $chunk = array_slice($docIds, $i, $chunkSize);
   $in = _srch_sql_in_numbers($chunk);

   $sql = "SELECT TokenID, DocID, TermFreq, FieldMask, Positions
         FROM Search_Posting_Table
         WHERE TokenID = " . (int)$tokenId . " AND DocID IN ($in)";

   $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.postingsL');

   foreach ($rows as $r) {
      if (!is_array($r)) continue;

      $out[] = [
         'TokenID'   => isset($r['TokenID'])   ? (int)$r['TokenID']   : (int)$tokenId,
         'DocID'     => isset($r['DocID'])     ? (int)$r['DocID']     : 0,
         'TermFreq'  => isset($r['TermFreq'])  ? (int)$r['TermFreq']  : 0,
         'FieldMask' => isset($r['FieldMask']) ? (int)$r['FieldMask'] : 0,
         'Positions' => isset($r['Positions']) ? (string)$r['Positions'] : '',
      ];
   }
```

```
    }

    return $out;
}

function _srch_load_docs_meta($docIds) {
    // Purpose: Load document metadata rows from Search_Doc_Table for a set of DocIDs.
    // NOTE: getRecordsFullSQL requires connectionType = 'company' because these search/index
    // tables are stored in the company database.

    $map = [];
    if (empty($docIds)) return $map;

    $chunkSize = 900;
    $docIds = array_values(array_unique($docIds));

    for ($i = 0; $i < count($docIds); $i += $chunkSize) {

        $chunk = array_slice($docIds, $i, $chunkSize);
        $in = _srch_sql_in_numbers($chunk);
```

```
    $sql = "SELECT DocID, SourceType, SourceName, BookName, ChapterName,
SectionNumber, SectionName, Title, LastIndexed
            FROM Search_Doc_Table
            WHERE DocID IN ($in)";

    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.docMeta');

    foreach ($rows as $r) {
      if (!is_array($r) || !isset($r['DocID'])) continue;

      $docId = (int)$r['DocID'];

      $map[$docId] = [
        'SourceType'    => isset($r['SourceType'])    ? (string)$r['SourceType']    : '',
        'SourceName'    => isset($r['SourceName'])    ? (string)$r['SourceName']    : '',
        'BookName'      => isset($r['BookName'])      ? (string)$r['BookName']      : '',
        'ChapterName'   => isset($r['ChapterName'])   ? (string)$r['ChapterName']   : '',
        // Keep as string to preserve formatting like 140.000
        'SectionNumber' => isset($r['SectionNumber']) ? (string)$r['SectionNumber'] : '',
        'SectionName'   => isset($r['SectionName'])   ? (string)$r['SectionName']   : '',
        'Title'         => isset($r['Title'])         ? (string)$r['Title']         : '',
        'LastIndexed'   => isset($r['LastIndexed'])   ? (string)$r['LastIndexed']   : '',
      ];
```

```
    }
  }

  return $map;
}

function _srch_load_screen_texts($names) {
  // Purpose: Load HelpMode + ScreenHelp for the matched ScreenName values.
  // NOTE: ScreenName values include '_' and must not be token-normalized.
  // NOTE: getRecordsFullSQL requires connectionType = 'company' because these tables
  // are stored in the company database.

  $map = [];
  if (empty($names)) return $map;

  $names = array_values(array_unique($names));
  $in = _srch_sql_quote_list($names);

  $sql = "SELECT ScreenName, HelpMode, ScreenHelp
      FROM Sec_Screen_Table
      WHERE ScreenName IN ($in)";
```

```
    $rows = getRecordsFullSQL($sql, 'company', 'searchExecute.scrText');

    foreach ($rows as $r) {
        if (!is_array($r) || !isset($r['ScreenName'])) continue;

        $name = (string)$r['ScreenName'];

        $map[$name] = [
            'HelpMode' => isset($r['HelpMode']) ? (string)$r['HelpMode'] : '',
            'Text'     => isset($r['ScreenHelp']) ? (string)$r['ScreenHelp'] : '',
        ];
    }

    return $map;
}

function _srch_load_proc_texts($names) {
    // Purpose: Load HelpMode + ProcedureText for the matched ProcedureName values.
    // NOTE: ProcedureName values may include '_' and must not be token-normalized.
    // NOTE: getRecordsFullSQL requires connectionType = 'company' because these tables
    // are stored in the company database.
```

```
$map = [];
if (empty($names)) return $map;

$names = array_values(array_unique($names));
$in = _srch_sql_quote_list($names);

$sql = "SELECT ProcedureName, HelpMode, ProcedureText
      FROM Proc_Procedure_Table
      WHERE ProcedureName IN ($in)";

$rows = getRecordsFullSQL($sql, 'company', 'searchExecute.procText');

foreach ($rows as $r) {
   if (!is_array($r) || !isset($r['ProcedureName'])) continue;

   $name = (string)$r['ProcedureName'];

   $map[$name] = [
      'HelpMode' => isset($r['HelpMode']) ? (string)$r['HelpMode'] : '',
      'Text'     => isset($r['ProcedureText']) ? (string)$r['ProcedureText'] : '',
   ];
}
```

```
    return $map;
}

function _srch_plain_text($html) {
    // Purpose: Convert stored HTML/text help content into normalized plain text
    // suitable for snippet extraction (preserve basic line breaks).

    if ($html === null) $html = '';
    if (!is_string($html)) $html = '';

    $s = html_entity_decode($html, ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
    if (!is_string($s)) $s = '';

    // Preserve basic structure as newlines before stripping tags
    $r = preg_replace('#<\s*br\s*/?>#i', "\n", $s);  $s = is_string($r) ? $r : '';
    $r = preg_replace('#</\s*p\s*>#i', "\n", $s);    $s = is_string($r) ? $r : '';

    $s = strip_tags($s);
    if (!is_string($s)) $s = '';

    // Normalize whitespace
```

```
    $r = preg_replace('/\r\n|\r|\n/u', "\n", $s);    $s = is_string($r) ? $r : '';
    $r = preg_replace('/[ \t]+/u', ' ', $s);        $s = is_string($r) ? $r : '';
    $r = preg_replace('/\n{3,}/u', "\n\n", $s);     $s = is_string($r) ? $r : '';

    return trim($s);
}

function _srch_make_snippet($rawHtml, $qTokens, $preferHitLines, $fallbackLines, $maxLen) {
    // Purpose: Build a short plain-text snippet favoring lines that contain any query token.

    $txt = _srch_plain_text($rawHtml);
    if ($txt === '') return '';

    $lines = preg_split('/\n/u', $txt);
    $lines = array_map('trim', $lines);
    $lines = array_values(array_filter($lines, function($l){ return $l !== ''; }));

    $hits = [];
    if ($preferHitLines > 0 && !empty($qTokens)) {

        // Query tokens are already normalized a-z0-9; build lookup for fast checks
        $qSet = [];
```

```
    foreach ($qTokens as $t) {
        $t = strtolower((string)$t);
        if ($t !== '') $qSet[$t] = true;
    }

    foreach ($lines as $ln) {

        // Tokenize the line the same way as the query so matching is consistent
        $lnTokens = _srch_tokenize($ln);

        $found = false;
        foreach ($lnTokens as $lt) {
            if (isset($qSet[$lt])) { $found = true; break; }
        }

        if ($found) {
            $hits[] = $ln;
            if (count($hits) >= $preferHitLines) break;
        }
    }
}
```

```
    $pick = !empty($hits) ? $hits : array_slice($lines, 0, (int)$fallbackLines);

    $snippet = trim(implode(' ', $pick));

    if (mb_strlen($snippet, 'UTF-8') > $maxLen) {
        $snippet = mb_substr($snippet, 0, $maxLen, 'UTF-8') . '...';
    }

    return $snippet;
}

function _srch_log_query($queryText, $results, $latencyMs) {
    // Purpose: Log a search query execution into Search_QueryLog_Table for analytics.

    // Optional user id from session if available (support both common keys)
    $userId = null;
    if (isset($_SESSION['loggedonuserid'])) {
        $userId = (string)$_SESSION['loggedonuserid'];
    } elseif (isset($_SESSION['UserID'])) {
        $userId = (string)$_SESSION['UserID'];
    }
```

```
    $qid = (int)getMaxValue('Search_QueryLog_Table', 'QueryID', '1=1', 'company') + 1;

    $fields = [
        'QueryID'   => $qid,
        'QueryText' => (string)$queryText,
        'UserID'    => $userId,
        'Results'   => (int)$results,
        'LatencyMs' => (int)$latencyMs,
        'QueryDate' => date('Y-m-d H:i:s'),
    ];

    createRecord('Search_QueryLog_Table', $fields, 'company');
}

?>
```

Appendix 9 – Model GUI front-end program – Code

In order to maintain uniformity within the entire program set of the system a model program is used for the front end GUIs. This is the sample code.

```
<?php

/*=== Document Identity ===

DocType: Program

Category: Main System Menu

Subsystem: Menu and Navigation

ProgramName: Sample.php

Audience: Both

Overview: Displays the main menu after user login, showing available modules and options based on user permissions. Provides navigation links to core system functions and serves as the central hub for accessing different parts of the application.

Keywords: menu, main menu, navigation, user permissions, modules, dashboard

=== End Document Identity === */

$debug="N";

require_once(getcwd()."/include/A_DBUserVariables.php") ;

require_once(getcwd()."/include/A_DBUserRoutines.php") ;

require_once(getcwd()."/include/A_CommonRoutines.php") ;

require_once(getcwd()."/include/startup.php") ;


// MAIN GLOBAL STATEMENT
```

```
global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
$accessLevelRequiredNum, $accessMessage;

global $secLevel, $secLevelRequired, $pctSecurityLevel,
$groupNumber,$moduleAccessLevel, $pctReadOK, $pctWriteOK;

global $sesPreFix, $sesPrefix,$activePane, $pctActivePane,$paneID;

global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;

global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;

global $updsw, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;


// ------------------------------------------------------------

// Screen identity k Replace Sample with screenmae ie SN_ABC

// replace XXX prefix with screen prefix ie abberiation for this screen like SN_ABC = SA,
SN_DEF = SD, etc.  This keeps session keys shorter and consistent

// set security parameters for this screen (group number, access level required, security
level required) - these will be used in the security check below and should be set based on
the needs of this screen and the user roles that should have access

// ------------------------------------------------------------

$screenName  = getCleanScreenName();

$screenTitle = "Sample";

$HELP_KEY = $screenName;

$paneID      = "Sample";$functionName = $paneID;

$sesPrefix  = "XXX";$sesPreFix  = $sesPrefix;

$groupNumber        = 1;    // security group number for this screen

$accessLevelRequired  = "R";  // required security level for this screen
```

```
$secLevelRequired    = "5";  // required security level for this screen



// ------------------------------------------------------------

// User identity

// ------------------------------------------------------------

// Your session dump shows: $_SESSION['loggedonuserid'] is the working key.

// NewpctUserID was being used without being set in this snippet.

// Set it from session so $pctUserid / $pctLoggedOnUserID are not blank.


//$NewpctUserID = $_SESSION['loggedonuserid'] ?? '';


$pctUserID       = $_SESSION['loggedonuserid'] ?? '';

$pctUserid       = $pctUserID;

$pctLoggedOnUserID = $pctUserID;


// ------------------------------------------------------------

// CHECK SECURITY

// ------------------------------------------------------------


// Step 1: Init minimum access based on module-wide needs

$moduleAccessLevel = initSecurityMinimum($accessLevelRequired);

$pctSecurityLevel  = $moduleAccessLevel;
```

```
if ($moduleAccessLevel == "N") { $pctReadOK = false; $pctWriteOK = false;
$accessMessage = "You are not authorized for this function"; }

if ($moduleAccessLevel == "R") { $pctReadOK = true;  $pctWriteOK = false;
$accessMessage = "You are authorized read access for this function"; }

if ($moduleAccessLevel == "W") { $pctReadOK = true;  $pctWriteOK = true;
$accessMessage = "You are authorized update access for this function"; }


// Step 2: Group-based fine security check

$accessLevel = $_SESSION['secgroups_array'][$groupNumber] ?? 'N';

$secLevel    = $_SESSION['seclevel'];


// Map access levels to numeric ranks: N < R < W

$accessLevelNum = 0;

if ($accessLevel === "R") { $accessLevelNum = 1; }

if ($accessLevel === "W") { $accessLevelNum = 2; }


$accessLevelRequiredNum = 0;

if ($accessLevelRequired === "R") { $accessLevelRequiredNum = 1; }

if ($accessLevelRequired === "W") { $accessLevelRequiredNum = 2; }


// Declining security: 1 is highest privilege, so user must be <= required

$accessAllow = "N";
```

```
if ($accessLevelNum > 0 && $accessLevelNum >= $accessLevelRequiredNum &&
$secLevel <= $secLevelRequired) {
   $accessAllow = "Y";
}

// ------------------------------------------------------------
// switches
// ------------------------------------------------------------
$updsw     = "N";
$addsw     = "N";
$btnhide   = "N";
$pksw      = "N";
$pctreturn  = "N";  // Return button switch (Y = show button after save/add)

//  all program globals should be initialized above this line for security and clarity

global $pctCSRFToken; // Make the CSRF token available globally for use in forms and
validation

// post back handling
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
  // fetch post fields
  //
  $updsw = "N";
  $updsw = isset($_POST['updsw']);

   // clear error message
   if (!isset($_SESSION['errmsg'])) {
      $_SESSION['errmsg'] = '';
   } else {
      $_SESSION['errmsg'] = ''; // Clear existing messages on each load
   }

   // issets for screen Sample

   if (isset($_POST['pctFieldName'])) { $pctFieldName = trim($_POST['pctFieldName']);
handleSessionNS($sesPreFix, 'set','pctFieldName',$pctFieldName); }



   $errsw  = false; // error switch to prevent processing if validation fails

   // Valiate CSRF token on form submission
if (!isset($_POST['csrfToken']) || $_POST['csrfToken'] !== $_SESSION['csrfToken']) {
```

```
    addMessage("Security validation failed. Please refresh and try again.");
    $errsw = true;
}

    /* sample button handler */
    if (isset($_POST['btn_Sample']))
    {
        If (!errsw)
        {
            $ok = ValidateSample();
            if (!ok)
            {
                $errsw = true;
                addMessage("Validation failed for Sample. Please check your input. ");
            }
        }
        If (!errsw)
        {
            // check security for update (MUST be first) group W R N with sec level 1-5
            if (!requireGroupAccess(4, "W", 3)) {$errsw = true;}

            $ok = ProcessSample();
```

```
        if (!ok)
        {
           $errsw = true;
           addMessage("Processing failed for Sample. Please check your input. ");
        }
      }

  }  // end of sample button handler */

} //End of post back handling


function validateSample()
{
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
$accessLevelRequiredNum, $accessMessage;
global $secLevel, $secLevelRequired, $pctSecurityLevel,
$groupNumber,$moduleAccessLevel, $pctReadOK, $pctWriteOK;
global $sesPreFix, $sesPrefix,$activePane, $pctActivePane,$paneID;
global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
global $updsw, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;
```

```
// additional globals here


// set error array for validation errors
$errs = [];


   // ---------------------------------------------------------------
   // Normalize session prefix (Ivan standard: SWU, no dash)
   // ---------------------------------------------------------------
   if (empty($sesPreFix) && !empty($sesPrefix))
      {  $sesPreFix = "XXX"; }


   // ---------------------------------------------------------------
   // Process validation for Sample screen fields here. Example shown for a numeric field; adjust as needed for your specific fields and validation rules.
   // ---------------------------------------------------------------
   $pctFieldName = (int)$pctFieldName; // example of casting to int for numeric field, adjust as needed for your field types
   if ($pctFieldName <= 0) {
      $errs[] = "FieldName  is missing. Cannot update.";
   }


   // text for errors set up message and exit
```

```
    // ---------------------------------------------------------------
    // Final result
    // ---------------------------------------------------------------
    if (!empty($errs)) {
        foreach ($errs as $msg) {
            addMessage($msg);
        }
        return false;
    }

    // validateion passed
    // reset sessoin vars from globals
    handleSessionNS($sesPreFix,'set','pctFieldName',$pctFieldName);

    return true;

}

function processSample()
```

```
{
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
$accessLevelRequiredNum, $accessMessage;
global $secLevel, $secLevelRequired, $pctSecurityLevel,
$groupNumber,$moduleAccessLevel, $pctReadOK, $pctWriteOK;
global $sesPreFix, $sesPrefix,$activePane, $pctActivePane,$paneID;
global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;
global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;
global $updsw, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;

// additional globals here


   // ---------------------------------------------------------------
   // Normalize session prefix (Ivan standard: SWU, no dash)
   // ---------------------------------------------------------------
   if (empty($sesPreFix) && !empty($sesPrefix)) {  $sesPreFix = "XXX"; }

   $errsw = false; // error switch to prevent processing if validation fails

    // make sure all globals are sent here for processing :
```

```
    // call createTableName_Table UpdateTableName_Table($keys)
deleteTableName_Table($keys) functions for processing here with appropriate keys and
values based on the needs of this screen and the database structure. Example shown for
update with a single key; adjust as needed for your specific processing needs.
    // test for error on return of update
    $errsw = true; // example of setting error switch to true if there was an error in processing,
adjust logic as needed based on your processing functions and error handling
    if (!$errsw)
    {
        addMessage("Process Completed Successfully."); // example of success message,
adjust as needed

    } else
    {
        addMessage("An error occurred during processing. Please check your input and try
again."); // example of error message, adjust as needed
        $errsw = true;
    }

}

function generateSampleHTML()
{
```

```
    global $accessLevel, $accessLevelRequired, $accessAllow, $accessLevelNum,
$accessLevelRequiredNum, $accessMessage;

global $secLevel, $secLevelRequired, $pctSecurityLevel,
$groupNumber,$moduleAccessLevel, $pctReadOK, $pctWriteOK;

global $sesPreFix, $sesPrefix,$activePane, $pctActivePane,$paneID;

global $screenName, $screen, $screenTitle, $functionName, $HELP_KEY;

global $NewpctUserID, $origPctUserID, $pctLoggedOnUserID, $pctUserID, $pctUserid;

global $updsw, $addsw, $btnhide, $pksw, $pctreturn, $pctReturnTo;

global $pctCSRFToken; // Make the CSRF token available globally for use in forms and
validation

$pctCSRFToken = handleSessionNS($sesPreFix, 'get', 'pctCSRFToken',
(string)($_SESSION['csrfToken'] ?? ''));


// All Screen globals Here


// Reset Globals for Session vars here

$pctFieldName = handleSessionNS($sesPreFix, 'get', 'pctFieldName', $pctFieldName ?? '');


 $html  = "";

   $html .= "<div class='property-pane' id='Sample' style='display:block;'>";

   $html .= "<form method='post'>";

   // When generating the  form, include the CSRF token as a hidden input

$html .= '<input type="hidden" name="csrfToken" value="' .
htmlspecialchars((string)($_SESSION['csrfToken'] ?? ''), ENT_QUOTES) . '">';
```

```
$html .= "<div style='text-align:center; padding:30px 10px;'>";

$html .= "<div style='margin-bottom:12px;'>";

$html .= "<img src='/images/BookBrandingLogo.png' alt='Logo' style='max-height:130px;
width:auto;'>";

$html .= "</div>";

$html .= "<div style='font-size:18px; color:#2596be; font-weight:600; margin-
bottom:6px;'>";

$html .= "Welcome to the Demo Website for";

$html .= "</div>";

$html .= "<div style='font-size:26px; color:#2596be; font-weight:700; margin-
bottom:6px;'>";

$html .= "Production Retrieval Augmented AI Systems";

$html .= "</div>";

$html .= "<div style='font-size:18px; color:#2596be; font-weight:600;'>";

$html .= "From Data Ingestion to LLM-Driven Applications";

$html .= "</div>";

$html .= "</div>";
```

```
    $html .= "</form>";
    return $html;

}


// Security check disallow allowAccess Y or N
function generateSecurityHTML()
{
    global $accessLevel, $secLevel, $accessAllow;
    $html = "";



    if ($accessAllow === "Y") {
      // ✅ Authorized — return empty or nothing
      return "";
   } else {
      // ❌ Not authorized — show warning message
      $html  = "<div style='margin: 2rem auto; padding: 2rem; max-width: 600px;";
      $html .= " background-color: #fff3cd; border: 1px solid #ffeeba;";
      $html .= " border-radius: 8px; color: #856404; font-size: 1.1rem;'>";
```

```
    $html .= "  <strong> 🚫 You are not authorized for this function.</strong><br><br>";
    $html .= "  Please select another option from the menu.<br><br>";
    $html .= "  <em>Your Access Level:</em> <strong>$accessLevel</strong><br>";
    $html .= "  <em>Your Security Level:</em> <strong>$secLevel</strong><br>";
    $html .= "  <em>Access Allowed:</em> <strong>$accessAllow</strong><br>";
    // new access level statement
    global $accessLevelRequired, $secLevelRequired;
    $html .= "Access Level required is " .$accessLevelRequired." and Security Level required
is " .$secLevelRequired. ".<br>";

    $html .= "</div>";
    return $html;
 }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title><?php echo $screenTitle; ?></title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <meta name="description" content="<?php echo $screenTitle; ?>">
    <meta name="keywords" content="<?php echo $screenTitle; ?>">
    <! —   Link css styles     —>
    <link rel="stylesheet" type="text/css" href="style/header.css" />
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

    <script src="/javascripts/writehtml.js"></script>

    <?php ListStyle(); ?>

<style>
/* 📊 Grid Table Styling */
.grid-table {
    width: 100%;
    border-collapse: collapse;
    font-size: 10pt;
    font-family: Arial, sans-serif;
}
.grid-table th, .grid-table td {
```

```
  border: 1px solid #ccc;
  padding: 4px 6px;
  text-align: left;
}
.grid-table th {
  background-color: #e3f2fd; /* Light blue header */
  color: #000;
}
.grid-table tr:hover {
  background-color: #f1f1f1;
  cursor: pointer;
}
</style>
<style>
  .lease-pane {
  background-color: #e3f2fd;      /* Light blue background */
  padding: 16px;                /* Comfortable padding */
  border-radius: 10px;          /* Rounded corners */
  border: 1px solid #ccc;        /* Light gray border */
  margin-top: 12px;             /* Space above each pane */
  min-height: 90vh;             /* 90% of screen height */
  display: none;                /* Start hidden, shown via JS */
```

```
}

.lease-pane h4 {
text-align: center;
font-weight: bold;
color: #1976d2; /* Deeper blue for header */
margin-bottom: 12px;
}
.tab-btn {
padding: 6px 12px;
margin-right: 6px;
border: none;
background-color: #bbdefb;
border-radius: 8px;
font-size: 10pt;
color: #000;
}

.tab-btn.active {
background-color: #1976d2;
color: white;
font-weight: bold;
```

```
  }


</style>
<style>
 /* apply lease look to property panes */
 .property-pane{
  background-color:#e3f2fd;    /* light blue */
  padding:16px;
  border-radius:10px;
  border:1px solid #ccc;
  margin:12px auto 0;
  max-width:1100px;          /* keeps content centered */
  display:none;              /* JS toggles to block */
  min-height:90vh;
 }
 .property-pane h5{
  text-align:center;
  font-weight:600;
  color:#1976d2;             /* header blue */
  margin-bottom:12px;
 }
```

```
 /* toolbar (property name + buttons) */
 .prop-toolbar{
  width:100%;
  margin:8px 0 12px 0;
  display:flex;
  justify-content:center;      /* <- center everything */
  align-items:center;
  gap:12px;
  flex-wrap:wrap;
 }
 .prop-toolbar label{ margin:0; }
 .prop-toolbar input[type="text"]{ width:30ch; max-width:100%; }
</style>



</head>
<body>

<!-- Header -->
<?php writePageHeader(true); ?>
<div class="container mt-4">  <!-- Main Body Division -->
```

```
   <center><input type='hidden' maxlength='1' size='1' name='updsw' readonly='readonly'
value='<?php echo $updsw; ?>'>

      <?php if (isset($_SESSION['errmsg'])) {echo '<span style="color: red;">' .
$_SESSION['errmsg'] . '</span>';unset($_SESSION['errmsg']); } ?>

   </center>

<!-- Dropdown Menu section -->

<?php ListMainMenu(); ?>


<?php

// check security disallow if needed

echo $html = generateSecurityHTML();

?>


<!-- Main Display-->

<div class="container mt-3">  <!-- Start of div class container mt 3 -->


 <!-- 🔒 Main Pane Section -->


<?php

if ($accessAllow === "Y")

{

$functionName = "generateSampleHTML";

echo '<div class="property-screen">';
```

```
$html = $functionName();

echo $html;

echo '</div>';

}

?>


</div> <!-- Main Display End-->


<!-- One-time modal block (already in your pattern) -->

<div id="helpModal" style="display:none; position:fixed; top:10%; left:10%; width:80%;
background:#fff; border:2px solid #ccc; padding:20px; z-index:1000;">

 <div id="helpModalBody" style="max-height:500px; overflow-y:auto; font-
size:0.95rem;"></div>

 <div style="text-align:right; margin-top:10px;">

  <button onclick="document.getElementById('helpModal').style.display='none'"
class="btn btn-secondary btn-sm">Close</button>

 </div>

</div>


<script>

  function copyGridToClipboard(tableId) {

    const table = document.getElementById(tableId);

    if (!table) {
```

```
        alert("Table not found.");
        return;
    }

    let text = '';
    for (let row of table.rows) {
        let rowText = [];
        for (let cell of row.cells) {
        rowText.push(cell.innerText.trim());
        }
        text += rowText.join('\t') + '\n';  // Tab-separated
    }

    navigator.clipboard.writeText(text).then(() => {
        alert("Entire grid copied to clipboard!");
    }).catch(err => {
        alert("Error copying grid: " + err);
    });
    }
</script>
```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></scri
pt>



<br>

</body>

</html>
```

Appendix 10 – Demo Web Site

A demonstration web site has been developed in order to show how all of the items presented in this book work. You can reach it at www.companyiadata.com . Use the button on the upper right to register, you will be asked for your name and email address. Once registered an email will be sent to you with your guest userid and password. Then again in the upper right corner you can press login to logon to the system. All screens of the system are visible to the you, however updates to the database are not allowed. We encourage you to use the Chat AI Agent. The system is loaded with a sample Human Resources Manual, an Accounting Practices Manual and a IT Policy and Procedures Manual.

Provided only as an example are some of the real questions and answers provided by the system.

Human Resources

Question

What is my companies overtime policy

DB Stream Response

Your company's overtime policy, based on the HR policy manual provided, is:

1. Overtime threshold
- Overtime applies to full-time employees for time worked over 8 hours in a day or over 44 hours in a week.

2. How overtime is handled
- Instead of immediate overtime pay, overtime hours are banked as lieu time at 1.5 times the actual hours worked.
- Example: if you work 2 overtime hours, you bank 3 hours of time off.

3. When lieu time is taken

- Banked time must be taken at a time mutually agreed upon by the ED/CEO and the employee.

4. If banked time is not used
- If the accumulated time is not taken within 6 months, the employer will pay those overtime hours at 1.5 times the employee's regular wage.

5. Daily maximum for certain travel/conference days
- The maximum claimable work time is 12 hours per day for out-of-town trips or conferences.

6. Approval requirement
- Employees must be authorized in advance to work overtime in order to receive credit for it.
- The exception is an emergency situation where prior approval cannot be obtained; in that case, the employee must advise the ED/CEO as soon as possible.

7. Required form
- Employees are required to submit a completed Overtime Approval Form to the ED/CEO for approval.

Important note:
- The same policy section says it needs to align with provincial employment standards and may be adjusted to fit the Foundation's practices. So this appears to be the current documented policy text, but it may be a template that should be confirmed against your final company handbook or province-specific rules.

Public Stream Response

In general, overtime policies usually define:
- when overtime starts, often after a daily or weekly threshold,
- whether overtime is paid or converted to compensatory time,

- the overtime rate, commonly 1.5x regular pay,
- whether pre-approval is required,
- any caps on claimable hours, and
- the forms or timesheets needed for approval and payroll.

Your internal policy specifically states overtime starts after 8 hours in a day or 44 hours in a week for full-time employees, is banked at 1.5x as lieu time, and must be pre-authorized except in emergencies.

Question

How many hours in a week do i have to work to be eligible for overtime

DB Stream Response

Based on the internal HR manual provided, there is no specific number of weekly hours stated for overtime eligibility. The manual says the Foundation should refer to its provincial employment standards legislation for hours of work and shows blanks for the standard workday and workweek, so the overtime threshold has not been filled in there.

The strongest internal answer is: your overtime eligibility depends on the province and the applicable employment standards rules, plus your employer's completed policy.

The document does point to provincial labour standards as the source to use. One selected section references Nova Scotia labour standards, which suggests Nova Scotia may be relevant, but the internal material still does not give the actual overtime threshold.

If you want, I can help you find the exact overtime threshold once you tell me your province and whether you are hourly, salaried, or in a management/exempt role.

Public Stream Response

Overtime eligibility is usually based on provincial or territorial employment standards law, not just your employer's handbook.

In Canada, the weekly overtime threshold depends on where you work. For example, many jurisdictions use 44 hours in a week, while others use different daily or weekly rules. Some employees are also exempt from overtime rules, such as certain managers or specific professions.

Because your internal document does not list the number, the accurate next step is to confirm:
1. Your province or territory
2. Whether your job is provincially or federally regulated
3. Whether your role is exempt from overtime

If you are in Nova Scotia, the internal material points to the Nova Scotia Labour Standards Code as the governing source, but it does not include the threshold itself.

Tell me your province, and I can give you the exact overtime-hours rule.

Question

Can i use my overtime and comp time for administrative leave

DB Stream Response

Based on the internal HR policy text provided:

1. Overtime / comp time
- Yes, the policy does allow overtime to be taken as banked time off rather than immediate

pay.
- Overtime hours are banked at 1.5x.
- That time off must be taken at a time mutually agreeable to the ED/CEO and the employee.
- If the banked time is not used within 6 months, the employer will pay it out at 1.5x the employee's regular wage.
- Overtime must be authorized in advance, except in emergencies.

2. Leave / administrative leave
- I do not see a clear internal policy here that gives a general right to use "administrative leave" the way you can use banked overtime/comp time.
- The leave section provided mentions unpaid emergency leave, possible unpaid personal leave depending on business needs, and medical/disability leave.
- The text does not clearly define a separate employee-use category called administrative leave.

So the strongest internal answer is:
- You likely can use your banked overtime/comp time, but only with approval and scheduling agreement with the ED/CEO.
- I cannot confirm from the provided policy that you can choose to use administrative leave in the same way.

If you want, I can help you draft a message to HR or your supervisor asking whether your banked lieu time can be used for a specific day and whether administrative leave is a separate approved category at your company.

Public Search Response

In general:

- Comp time or lieu time can usually be used if your employer's policy allows it, and often it

must be approved in advance and scheduled based on operational needs.
- Administrative leave is different. Employees usually cannot just elect to use administrative leave on their own unless company policy specifically provides for it. Administrative leave is often granted by the employer for specific situations.

So in plain terms:
- Comp time: often yes, if you have it banked and your supervisor approves the time.
- Administrative leave: usually only if your employer authorizes it under a defined policy.

If you tell me what kind of day off you need, I can help you figure out which leave type fits best.

IT Policy

Question

What is my company anti-virus policy

DB Stream Response

Based on the internal policy excerpts provided, your company's anti-virus policy appears to require that internet-connected devices be protected with up-to-date antivirus software.

What the policy says:
1. Antivirus is defined as software used to prevent, detect, and remove malicious software.
2. Employees should make sure their computer or device has the latest antivirus software.
3. Employees should also turn on the firewall.
4. Employees should avoid downloads on public Internet connections.
5. Employees should use strong passwords.
6. For laptops, tablets, phones, and similar devices, employees are expected to defend them against Internet threats.

Practical takeaway:
- Keep antivirus installed and updated on your laptop, tablet, phone, or other internet-

connected device.
- Keep the firewall enabled.
- Do not download files while on public Internet connections.
- Use strong passwords.
- Be cautious with Wi-Fi and use trusted, secure networks.

Limit note: The excerpts provided do not show a full anti-virus policy covering items like approved antivirus products, scan frequency, patch enforcement, reporting steps, or exceptions. If you want, I can help you turn this into a short formal policy summary for employees.

This response may not fully satisfy your request. Try rephrasing your question or using related keywords.

Public Stream Response

A typical company anti-virus policy usually requires employees to:
1. Install and maintain approved antivirus or endpoint protection software on company devices.
2. Keep virus definitions and software updates current.
3. Enable real-time protection and firewalls.
4. Avoid unsafe downloads, especially on public or untrusted networks.
5. Report suspected malware or suspicious activity to IT or security immediately.
6. Avoid disabling security tools without authorization.

In your provided internal material, the clearly supported requirements are to keep the latest antivirus software on devices, enable the firewall, avoid downloads on public Internet connections, use strong passwords, and defend internet-connected devices against threats.

Question

what is the policy about confidential data

DB Stream Response

Based on the selected internal policy content, the policy on confidential data is:

1. Confidential information must always be treated with care.
2. Employees must prevent unauthorized persons from accessing member information, applications, or data.
3. Employees must check their work environment to avoid any loss of confidentiality involving data media or documents.
4. PURPLESEC requires a clean desk approach for protecting confidential and sensitive information.

The policy specifically applies to items such as:
- Day planners and Rolodexes containing non-public information
- File cabinets, storage cabinets, and briefcases with sensitive or confidential information
- Reports, lists, and statements containing confidential or sensitive data
- Electronic devices such as cell phones and PDAs
- Keys used to access sensitive information
- Printouts containing sensitive information
- Data left on printers, copiers, or fax machines
- Computer workstations and passwords
- Portable media such as CDs, disks, and flash drives
- Desks and work areas, including whiteboards and bookshelves

The policy defines sensitive data to include personal information and restricted data. Personal information includes items such as a person's name, Social Security number, driver's license or ID number, and financial account or card information with access credentials.

Restricted data includes:
- Personal data that identifies and describes an individual
- Limited data whose unauthorized access, modification, or loss could seriously harm PURPLESEC, its members, or non-members

Additional related policy points from the selected material:
- PURPLESEC must protect the confidentiality, security, and integrity of members' non-public personal information.
- PURPLESEC maintains administrative, technical, physical, electronic, and procedural safeguards.
- Member information should only be collected when necessary for products, services, transactions, or other relevant business purposes.
- PURPLESEC does not sell or provide member information to third parties for independent use.
- Access to confidential information should be limited to those who need it for their job duties, and paper records should be locked while computer records are protected with passwords.

In short, the policy requires careful handling, restricted access, secure storage, clean desk practices, and appropriate safeguards for all confidential and sensitive data.

Public Stream Response

A typical policy about confidential data says that confidential or sensitive information must be protected from unauthorized access, use, disclosure, alteration, or loss.

Common requirements usually include:
- Only access confidential data if you need it for your job
- Do not share it for personal gain or with unauthorized people
- Keep paper records secured, such as in locked storage
- Protect electronic records with passwords and other safeguards

- Avoid leaving sensitive information unattended on desks, printers, whiteboards, or portable devices
- Use care when handling personally identifiable information, financial data, or other restricted data
- Follow clean desk and secure disposal practices
- Share information with third parties only when authorized and necessary

Confidential data often includes:
- Names and contact details
- Social Security numbers or government ID numbers
- Financial account or payment card information
- Passwords or access codes
- Sensitive business, donor, employee, customer, or member records

The overall goal of this kind of policy is to preserve confidentiality, security, privacy, and legal compliance.

Question

can i use corporate email for private use

DB Stream Response

Yes, but only in a limited and restricted way.

Based on the internal policy excerpts:
- PURPLESEC e-mail is to be used for business purposes.
- Using e-mail for conducting personal business is listed as prohibited.
- However, the policy also allows incidental personal use of sending e-mail for PURPLESEC approved users.

That incidental personal use is limited by these conditions:

1. It does not extend to family members or acquaintances.
2. It must not result in direct costs to PURPLESEC without prior management approval.
3. It must not interfere with your normal work duties.
4. No files or documents may be sent or received that could create legal liability for or embarrassment to PURPLESEC.
5. Storage of personal files and documents in PURPLESEC IT systems should be nominal.

Important policy warning:
- Corporate e-mail is not private.
- PURPLESEC may monitor e-mail without prior notification.
- All e-mails, files, and documents, including personal ones on company systems, are owned by PURPLESEC.
- Email usage is subject to logging and review.

So the practical answer is: limited incidental personal use may be allowed, but not personal business use, and you should assume there is no privacy.

Public Stream Response

In many organizations, limited personal use of corporate email is allowed, but only if it is occasional, does not interfere with work, does not create cost or legal risk for the employer, and does not involve outside business activity.

Common rules usually include:
- Do not use corporate email for running a side business or personal commercial activity
- Do not send offensive, harassing, or inappropriate content
- Do not store large amounts of personal material
- Do not send confidential company information to personal email accounts
- Assume corporate email is monitored and not private

So generally, occasional incidental personal use may be acceptable, but private business use usually is not.

Accounting

Question

How are spreadsheets use in accounting in my company

DB Stream Response

In your company, spreadsheets are used as an important part of the accounting and financial process.

Based on the policy provided, significant spreadsheets are used for three main purposes:

1. Financial reporting
- They help determine financial statement transaction amounts or balances that are later recorded in the general ledger.
- They are also used as a key control in financial reporting, such as balancing and reconciling significant accounts.
- They may create or support financial reports and disclosures.

2. Analytical use
- They support management decision-making by organizing and analyzing financial and business information.

3. Operational use
- They help track and monitor workflow, such as open claims, unpaid invoices, and other business information.

Your Finance Department is responsible for identifying spreadsheets that are significant to the financial process. It keeps a list of those spreadsheets, including their purpose, file location, and the owner or end user, and updates that list annually.

Your company also requires controls over significant spreadsheets to protect accuracy and

security:
- They must be saved on a secured corporate server, not a personal hard drive.
- Access is limited to employees with a legitimate business need.
- High-risk or sensitive spreadsheets may be password protected.
- Formula and calculation cells are locked.
- Changes must be logged, tested, described, and documented.
- Updated spreadsheets must show version number and publication date.
- Printed spreadsheets used for retention or grant documentation must include standard identifying headers and footers.

The policy also says that before developing complex spreadsheets for financial calculations or information

Public Stream Response

In many organizations, spreadsheets are commonly used in accounting to support work such as:

1. Financial reporting
- Preparing schedules for month-end and year-end reporting
- Supporting balances that feed the general ledger or financial statements
- Creating reconciliation workpapers

2. Analysis
- Budget-to-actual comparisons
- Trend analysis
- Forecasting and scenario modeling

3. Operational tracking
- Monitoring invoices, accruals, fixed assets, projects, or claims
- Maintaining supporting schedules for recurring accounting entries

Good practice is to control important accounting spreadsheets by restricting access, locking formula cells, documenting changes, using version control, and storing them in secure locations. Many companies also review whether a spreadsheet should be replaced by a more controlled financial system when it becomes too complex or high risk.

Question

what are my companies procurement procedures

DB Stream Response

Your company's procurement procedures, based on the internal policy provided from the City of Superior Accounting and Financial Policies, are:

1. Scope and authority
- These purchasing policies apply to all purchases made by the City of Superior.
- Purchases are assumed to be budgeted in a department's annual budget.
- Department Heads or their designees may initiate purchases for their department.
- Department Heads must inform Finance of who is authorized to initiate purchases.
- The Finance Department maintains the authorized purchaser list in the purchasing system.
- The Finance Director has approval authority over purchases and contractual commitments under the applicable authorization thresholds and can deny proposed purchases based on budgetary or other conditions.

2. Ethical conduct and conflicts
- Staff must follow ethical, responsible, and reasonable purchasing practices.
- Staff must discourage and decline gifts or gratuities that could influence purchasing decisions.
- Staff must notify their immediate supervisor if such gifts are offered.
- Officers, Council members, employees, and agents may not participate in contractor

selection or administration where a real or apparent conflict of interest exists.
- They also may not solicit or accept gratuities, favors, or anything of monetary value from contractors or subcontract parties.
- Unsolicited gifts valued at $25 or less may be accepted only with Department Head approval.

3. Competition requirements
To support open and full competition, purchasers must:
- Watch for internal conflicts of interest.
- Watch for noncompetitive practices among contractors.
- Not allow contractors that develop specifications, requirements, or proposals to bid on those same procurements.
- Award to the bidder whose product or service is most advantageous considering price, quality, and other factors.
- Issue solicitations that clearly state all evaluation requirements.
- Reserve the right to reject any and all bids if it is in the City's best interest.
- Avoid geographic preference unless required by federal statute.
- Use brand-name-or-equal descriptions only to define needed performance or requirements.

4. Nondiscrimination requirements for contractors
Contractors receiving City funds or providing goods or services under City agreements must:
- Not discriminate against employees or applicants because of race, religion, color, sexual orientation, or national origin, except where legally permitted as a bona fide occupational qualification.
- Post notices of the nondiscrimination clause in conspicuous places available to employees and applicants.

5. Core procurement procedures
The policy lists these procurement procedures:

- Avoid buying items that are unnecessary or duplicative for activities required by a federal award.
- When appropriate, analyze lease versus purchase options to determine the most economical and practical approach, but only when both options are available.
- Use state or local intergovernmental or inter-entity agreements where appropriate for shared goods and services.
- Use federal excess and surplus property instead of buying new equipment when feasible and cost-reducing.
- Keep documentation of cost and price analysis for each procurement decision over the simplified acquisition threshold of $150,000 in the procurement file for the federal award.
- Ensure pre-qualified vendor, firm, or product lists are current and contain enough qualified sources to maximize open and full competition.
- Maintain records showing the history of the procurement, including:
a. Rationale for the procurement method
b. Selection of contract type
c. Contractor selection or rejection
d. Basis for the contract price
- Make procurement files available for inspection upon request by a federal or pass-through awarding agency.
- Do not use cost-plus-a-percentage-of-cost or percentage-of-construction-cost contracting methods.

6. Service purchasing threshold noted in the selected policy
For services over $25,000:
- The responsible department head or designee must prepare a written request for proposals.
- The RFP must describe the services requested, information required, and submission deadline.
- Bid packets must include: Subcontractor and Suppliers Listing, Addenda Acknowledgment, Bidders Proof of Responsibility, and Affidavit of Organization and Authority.

- The bid must be advertised for two consecutive weeks.
- Sealed bids are opened at the deadline with at least two City staff members present.
- The department head prepares a report listing proposals received and the recommendation.
- The report goes to the common council, which decides which proposal, if any, to accept.
- If accepted, the department head notifies the successful party and the contract analyst prepares the contract.

Exceptions listed for service bid procedures:
- Purchases of services under $25,000.
- Certain demolition contracts.
- Emergency services when delay would materially harm City operations, finances, property, people, or position.
- Purchases made under a state contract bid.

7. Equipment, materials, and supplies bid exceptions noted in the selected policy
Exceptions include:
- Single purchases of equipment, materials, or supplies of $25,000 or less.
- Sole-source replacement parts for existing equipment.
- Certain patented equipment when it is the only satisfactory option.
- Equipment rentals without a purchase option.
- Public construction contracts governed by cited Wisconsin statutes.
- Used equipment purchases.
- Emergency purchases.
- Purchases under a state contract bid.

One more related compliance note from the selected material:
- Violations of this policy are treated as violations of the City’s Code of Ethics and are subject to investigative, reporting, and disclosure procedures under the misconduct policy.

If you want, I can turn this into a shorter SOP-style checklist or a one-page employee summary.

Public Stream Response

In general, procurement procedures typically cover these areas:

1. Authorization
- Define who can initiate purchases and who must approve them.
- Set approval thresholds by dollar amount and purchase type.

2. Ethical standards
- Prohibit conflicts of interest, improper gifts, kickbacks, and favoritism.
- Require staff to disclose potential conflicts.

3. Competition and sourcing
- Require quotes, bids, or proposals depending on the dollar threshold.
- Use clear specifications and fair evaluation criteria.
- Document why a vendor was selected.

4. Purchasing methods
- Small purchases may use simplified quote procedures.
- Larger purchases often require formal sealed bids or RFPs.
- Sole-source and emergency purchases usually require justification.

5. Documentation and recordkeeping
- Keep records of requisitions, quotes, bids, evaluations, approvals, contracts, and pricing analysis.
- Retain enough detail to show how the procurement decision was made.

6. Contracting controls

- Avoid prohibited pricing structures where required by law or policy.
- Ensure contracts include required legal, insurance, and compliance terms.

7. Compliance requirements
- Include nondiscrimination, grant, and regulatory requirements when applicable.
- Allow audits or inspections where funding rules require it.

Based on your internal documents, your organization appears to follow a structured public-sector procurement model with ethics rules, competition requirements, documentation standards, and special procedures for purchases over $25,000 and for federally funded purchases.

Question

how do i record accounts payable

DB Stream Response

Based on your internal accounting policies, accounts payable should be recorded like this:

1. Record only valid, supported payables
- All valid accounts payable transactions must be recorded in a timely manner.
- The entry should be supported by the required documentation.
- Amounts recorded are based on the contractor invoice for the related goods or services.

2. Make sure the invoice is properly approved first
- The contractor invoice should be reviewed and approved by the Department Head or Supervisor, depending on the amount, before it is processed for payment.
- Invoices or check requests sent to Finance must include the necessary approvals and supporting documentation.
- All vouchers should be signed off by the appropriate officials.

3. Include the required support in the payment package

Your policy indicates the voucher package should include:

- the invoice
- the voucher
- a receiving copy signed by the receiver
- vendor signature on the voucher if required over the applicable quote threshold
- documentation showing date of receipt and whether goods were received in full or in part

4. Enter the payable through the A/P process

- Accounts payable are processed weekly.
- Information is entered into the system from approved invoices or disbursement vouchers with documentation attached.
- Invoices and related general ledger account distribution codes are reviewed before posting to the subsidiary system.

5. Follow document control rules

- Only original invoices should be processed unless duplicate copies have been verified as unpaid by researching contractor records.
- Emailed invoices should be printed, date-stamped, and initialed by an accounting clerk.
- Extra copies of emailed invoices should be deleted.
- Do not make payments from contractor statements.

6. Apply segregation of duties

- The asset or expense and the related liability should be recorded by someone who is not responsible for ordering and receiving.

7. Apply month-end and year-end cutoff rules

- For monthly financial statements, invoices received, approved, and properly supported by the fifth day of the following month should be recorded as accounts payable at the end of the prior month if the goods or services were delivered by month-end.
- At year end, complete all accrual entries and review year-end purchases to determine

whether they should be recorded as accounts payable or as carry-forward encumbrances.

8. Pay timely and reconcile
- Verify invoices are paid in a timely manner.
- Reconcile accounts in a timely manner.

In short, your process is: receive invoice and support, confirm receipt of goods or services, obtain approvals, code the expense, enter it into the A/P system weekly, and record it in the proper accounting period based on delivery date and cutoff rules.

If you want, I can turn this into a simple step-by-step SOP checklist for entering an invoice.

Public Stream Response

In general, accounts payable is recorded when your organization has received goods or services and owes money to a vendor.

Typical process:
1. Receive the vendor invoice.
2. Match it to supporting documents such as a purchase order and receiving evidence.
3. Confirm the goods or services were actually received.
4. Obtain required approval.
5. Code the invoice to the correct expense or asset account.
6. Record the liability in accounts payable.
7. Pay the invoice according to approved terms.

Typical journal entry when recording the payable:
- Debit: Expense or Asset
- Credit: Accounts Payable

Example:

If you receive a $1,000 office supply invoice:

- Debit Office Supplies Expense $1,000
- Credit Accounts Payable $1,000

When you pay it later:

- Debit Accounts Payable $1,000
- Credit Cash $1,000

A key accounting rule is cutoff: if goods or services were received before period-end, the payable should usually be recorded in that period even if the invoice is processed shortly after month-end, subject to company policy.

Good practice also includes using original invoices, requiring approvals, avoiding duplicate payments, and separating duties between purchasing, receiving, recording, and payment.

Appendix 11 – Recommended Reading and Industry References

The following references represent selected books, papers, architectural discussions, and industry resources relevant to enterprise retrieval systems, Retrieval-Augmented Generation, search architecture, metadata engineering, graph-enhanced retrieval, and Artificial Intelligence operational design. These references influenced portions of the architectural thinking discussed throughout this manuscript and are recommended for readers interested in deeper exploration of enterprise knowledge systems and retrieval engineering.

1. Manning, Christopher D.; Raghavan, Prabhakar; Schutze, Hinrich
   Introduction to Information Retrieval
   Cambridge University Press

One of the foundational modern texts on information retrieval, indexing, ranking, scoring, search architecture, and retrieval system engineering. Particularly valuable for understanding the historical and technical foundations underlying enterprise search systems.

2. Jurafsky, Daniel; Martin, James H.
   Speech and Language Processing
   Pearson

Widely regarded as one of the most important references in Natural Language Processing. Covers statistical language processing, semantic analysis, embeddings, retrieval concepts, and computational linguistics relevant to modern AI systems.

3. Russell, Stuart; Norvig, Peter
   Artificial Intelligence: A Modern Approach
   Pearson

A broad Artificial Intelligence reference covering reasoning systems, search, machine learning, knowledge representation, and AI architecture concepts foundational to enterprise AI systems.

4. Lewis, Patrick; Perez, Ethan; Piktus, Aleksandra; et al.
   Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
   Meta AI Research Paper

One of the early influential papers formally describing Retrieval-Augmented Generation architectures and the integration of retrieval systems with large language models.

5. White, Tom
   Hadoop: The Definitive Guide
   O'Reilly Media

Important background reading for understanding large-scale data systems, distributed processing concepts, indexing infrastructure, and scalable information architectures relevant to enterprise knowledge systems.

6. Knuth, Donald E.
   The Art of Computer Programming
   Addison-Wesley

While not directly focused on AI retrieval, this foundational series strongly reinforces disciplined systems engineering, algorithmic thinking, indexing concepts, sorting methods, and computational architecture principles.

7. Hearst, Marti A.
   Search User Interfaces
   Cambridge University Press

A highly valuable reference discussing how users interact with retrieval systems, search behavior, navigation structures, filtering, and retrieval presentation design.

8. Chen, Wenhu; Wang, Hexiang; et al.
   Graph-Based Retrieval-Augmented Generation
   Industry Research Discussions and Emerging Publications Beginning Around 2024

A growing body of research discussing graph-enhanced retrieval architectures, relationship-aware retrieval systems, connected knowledge structures, and contextual reasoning across enterprise information environments.

9. Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S.
   Database System Concepts
   McGraw-Hill

A foundational reference covering relational database architecture, normalization, indexing, query systems, data modeling, and structured enterprise information management.

10. Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier
    Modern Information Retrieval
    Addison-Wesley

One of the classic references on information retrieval theory, ranking models, indexing, search architecture, relevance scoring, and retrieval system design.

The enterprise retrieval architectures discussed throughout this manuscript were also influenced by practical operational experience involving:

- enterprise accounting systems
- operational reporting environments
- spreadsheet-driven business analysis
- metadata-heavy document repositories
- procedural workflow systems
- search-engine engineering
- large-scale document ingestion pipelines
- hybrid retrieval experimentation
- relationship-aware retrieval architectures
- real-world enterprise operational constraints

The author encourages readers to explore both academic retrieval research and practical enterprise systems engineering literature. Many of the most important advancements in enterprise Artificial Intelligence emerge not from isolated AI models alone, but from the integration of retrieval engineering, operational architecture, metadata systems, governance structures, and contextual knowledge organization working together as unified systems.