INTERNSHIP REPORT

(21/05/2025 - 20/06/2025)

BHARGAV SAI KOLLIPARA

HU22CSEN0100620

COMPUTER SCIENCE AND ENGINEERING

GITAM HYDERABAD

Project Description

Request handler for Internal Network connections

This project focuses on building an internal server run web application making the work easy for the respective authorised personnel to keep the check and provide access for respective departments in the company with an access for their internal NETWORK.

The web-app consists of 4 pages

- Login page
- Normal user page(applies for connection)
- IT-Champion page(Checks and forwards the application to HoD)
- HoD page (Checks for compliance and approves/declines requests)

Tech stack :

HTML/CSS

For building all the appearance part of the web-app, built clean and interactive pages with comfortable dashboards which make the work easier for respective users of the page.

Javascript

The validation part of frontend was handled using integrated javascript with html.

.NET (C#)

The whole website's backend is developed and supported with .NET(C#) as all the web apps in use by the office staff and systems were in built in the traditional .NET platform.

MySQL

MySQL was used for manipulating the data.

GitHub

GitHub was used for version control while building the web-app.

AI-Tools

Artificial intelligence tool like Claude, ChatGpt, Gemini were used at their best to achieve a short time span in building this project.

At the end of this tenure as an intern at E.C.I.L I have learnt to build a web-app on the .net platform and a programming language C#.
I have learnt to efficiently use AI tools for work efficiency at my best.

Code used to build

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>INTERNET Login</title>
  <style>
    body {
        margin: 0;
        padding: 0;
        font-family: Arial, sans-serif;
        background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
        min-height: 100vh;
        display: flex;
        align-items: center;
        justify-content: center;
    }

    .login-container {
        background: rgba(255, 255, 255, 0.95);
        padding: 40px;
        max-width: 400px;
        width: 90%;
        border-radius: 15px;
        box-shadow: 0 15px 35px rgba(0, 0, 0, 0.1);
        backdrop-filter: blur(10px);
    }

    .logo {
        text-align: center;
        margin-bottom: 30px;
    }

    .logo h1 {
        color: #333;
        margin: 0;
        font-size: 28px;
        font-weight: bold;
    }

    .logo p {
        color: #666;
        margin: 5px 0 0 0;
        font-size: 14px;
    }

    .form-group {
```

```
    margin-bottom: 20px;
}

label {
    display: block;
    margin-bottom: 8px;
    color: #333;
    font-weight: 500;
}

input, select {
    width: 100%;
    padding: 12px;
    border: 2px solid #e1e1e1;
    border-radius: 8px;
    font-size: 16px;
    transition: border-color 0.3s ease;
    box-sizing: border-box;
}

input:focus, select:focus {
    outline: none;
    border-color: #667eea;
}

.login-btn {
    width: 100%;
    padding: 14px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    transition: transform 0.2s ease;
}

.login-btn:hover {
    transform: translateY(-2px);
}

.login-btn:disabled {
    background: #ccc;
    cursor: not-allowed;
    transform: none;
}

.error-message {
    background: #fee;
    color: #c33;
    padding: 10px;
```

```
      border-radius: 8px;
      margin-bottom: 20px;
      display: none;
    }

    .loading {
      display: none;
      text-align: center;
      color: #666;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="login-container">
    <div class="logo">
      <h1>INTERNET</h1>
      <p>Electronics Corporation of India Limited</p>
    </div>

    <div id="errorMessage" class="error-message"></div>

    <form id="loginForm">
      <div class="form-group">
        <label for="username">Username</label>
        <input type=text id="username" name="username" required>
      </div>

      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>
      </div>

      <div class="form-group">
        <label for="role">Select Role</label>
        <select id="role" name="role" required>
          <option value="" disabled selected>-- Choose Role --</option>
          <option value="NormalUser">Normal User</option>
          <option value="ITChampion">IT Champion</option>
          <option value="HOD">HOD</option>
        </select>
      </div>

      <button type="submit" class="login-btn" id="loginBtn">Login</button>
      <div class="loading" id="loadingMsg">Logging in...</div>
    </form>
  </div>

 <script>
    const API_BASE = ''; // Corrected: Set to empty string for direct ASHX calls

    document.getElementById('loginForm').addEventListener('submit', async function(e) {
```

```
e.preventDefault();

const username = document.getElementById('username').value;
const password = document.getElementById('password').value;
const role = document.getElementById('role').value; // Get the selected role
const errorDiv = document.getElementById('errorMessage');
const loadingDiv = document.getElementById('loadingMsg');
const loginBtn = document.getElementById('loginBtn');

// Reset error message
errorDiv.style.display = 'none';

// Show loading state
loadingDiv.style.display = 'block';
loginBtn.disabled = true;

try {
  // CORRECTED: Fetch call should go to LoginHandler.ashx
  // CORRECTED: Include 'role' in the JSON body
  const response = await fetch(`LoginHandler.ashx`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      username: username,
      password: password,
      role: role // Pass the selected role to the handler
    })
  });

  // Check if response is ok first
  if (!response.ok) {
    throw new Error('Invalid credentials or server error');
  }

  // Get response text first
  const responseText = await response.text();

  // Check if response is empty or not JSON
  if (!responseText || responseText.trim() === '') {
    throw new Error('Invalid credentials');
  }

  let data;
  try {
    data = JSON.parse(responseText);
  } catch (jsonError) {
    // If JSON parsing fails, it's likely an error response
    throw new Error('Invalid credentials');
  }
```

```
        // Check if login was successful
        if (data.success === false || data.error) {
          throw new Error(data.message || data.error || 'Invalid credentials');
        }

        // Check if the selected role matches the user's role
        if (data.Role !== role) {
          throw new Error('Selected role does not match your account role');
        }

        // Store user info
        sessionStorage.setItem('userInfo', JSON.stringify({
          username: data.Username,
          role: data.Role,
          userId: data.UserId
        }));

        // Redirect based on role
        switch (data.Role) {
          case 'NormalUser':
            window.location.href = 'normalUser.html';
            break;
          case 'ITChampion':
            window.location.href = 'it-champion.html';
            break;
          case 'HOD':
            window.location.href = 'hod.html';
            break;
          default:
            throw new Error('Unknown role');
        }
      } catch (error) {
        errorDiv.textContent = error.message;
        errorDiv.style.display = 'block';
      } finally {
        loadingDiv.style.display = 'none';
        loginBtn.disabled = false;
      }
    });

    // Test credentials info
    console.log('Test Credentials:');
    console.log('Normal User: user1 / pass123');
    console.log('IT Champion: champion1 / pass456');
    console.log('HOD: hod1 / pass789');
  </script>

</body>
</html>
```

NormalUser.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>ECIL - Internet Request Form</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
      margin: 0;
      padding: 20px;
      min-height: 100vh;
    }

    .header {
      text-align: center;
      background: white;
      padding: 20px;
      border-radius: 15px;
      box-shadow: 0 5px 15px rgba(0,0,0,0.1);
      margin-bottom: 30px;
    }

    .header h1 {
      color: #333;
      margin: 0;
      font-size: 28px;
    }

    .header p {
      color: #666;
      margin: 5px 0 0 0;
    }

    .user-info {
      background: #e8f4fd;
      padding: 15px;
      border-radius: 10px;
      margin-bottom: 20px;
      border-left: 4px solid #2196F3;
    }

    .form-container {
      background: white;
      padding: 40px;
      border-radius: 15px;
      box-shadow: 0 10px 30px rgba(0,0,0,0.1);
      max-width: 800px;
      margin: 0 auto;
```

```
}

.section-title {
  color: #333;
  border-bottom: 2px solid #2196F3;
  padding-bottom: 10px;
  margin-bottom: 25px;
  margin-top: 30px;
  font-size: 20px;
}

.section-title:first-child {
  margin-top: 0;
}

.form-row {
  display: flex;
  gap: 20px;
  margin-bottom: 20px;
}

.form-group {
  flex: 1;
  margin-bottom: 20px;
}

label {
  display: block;
  margin-bottom: 8px;
  color: #333;
  font-weight: 500;
}

input, select, textarea {
  width: 100%;
  padding: 12px;
  border: 2px solid #e1e1e1;
  border-radius: 8px;
  font-size: 16px;
  transition: border-color 0.3s ease;
  box-sizing: border-box;
}

input:focus, select:focus, textarea:focus {
  outline: none;
  border-color: #2196F3;
}

input[readonly] {
  background-color: #f8f9fa;
  color: #6c757d;
}
```

```
.checkbox-group {
  display: flex;
  align-items: flex-start;
  gap: 10px;
  margin-bottom: 20px;
  padding: 15px;
  background: #f8f9fa;
  border-radius: 8px;
}

.checkbox-group input[type="checkbox"] {
  width: auto;
  margin-top: 4px;
}

.checkbox-group label {
  margin-bottom: 0;
  font-size: 14px;
  line-height: 1.4;
}

.submit-section {
  text-align: center;
  margin-top: 40px;
  padding-top: 30px;
  border-top: 1px solid #eee;
}

.submit-btn {
  background: linear-gradient(135deg, #2196F3 0%, #1976D2 100%);
  color: white;
  padding: 15px 40px;
  border: none;
  border-radius: 8px;
  font-size: 16px;
  font-weight: 600;
  cursor: pointer;
  transition: transform 0.2s ease;
}

.submit-btn:hover {
  transform: translateY(-2px);
}

.submit-btn:disabled {
  background: #ccc;
  cursor: not-allowed;
  transform: none;
}

.message {
```

```
      padding: 15px;
      border-radius: 8px;
      margin-bottom: 20px;
      display: none;
    }

    .success {
      background: #d4edda;
      color: #155724;
      border: 1px solid #c3e6cb;
    }

    .error {
      background: #f8d7da;
      color: #721c24;
      border: 1px solid #f5c6cb;
    }

    .logout-btn {
      position: absolute;
      top: 20px;
      right: 20px;
      background: #dc3545;
      color: white;
      padding: 8px 16px;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      text-decoration: none;
      font-size: 14px;
    }

    @media (max-width: 768px) {
      .form-row {
        flex-direction: column;
      }

      .form-container {
        padding: 20px;
      }
    }
  </style>
</head>
<body>
  <button class="logout-btn" onclick="logout()">Logout</button>

  <div class="header">
    <h1>Internet Request Form</h1>
    <p>Electronics Corporation of India Limited</p>
  </div>

  <div class="user-info">
```

```
    <strong>Logged in as:</strong> <span id="currentUser"></span> (Normal User)
  </div>

  <div id="message" class="message"></div>

  <div class="form-container">
    <form id="requestForm">
      <h2 class="section-title">Particulars of the Applicant</h2>

      <div class="form-row">
        <div class="form-group">
          <label for="applicantName">Name of the Applicant</label>
          <input type="text" id="applicantName" name="applicantName" readonly>
        </div>
        <div class="form-group">
          <label for="divSecCode">Div-Sec-Code No.</label>
          <input type="text" id="divSecCode" name="divSecCode" required>
        </div>
      </div>

      <div class="form-row">
        <div class="form-group">
          <label for="designation">Designation</label>
          <input type="text" id="designation" name="designation" required>
        </div>
        <div class="form-group">
          <label for="contactPhone">Contact Phone No.</label>
          <input type="tel" id="contactPhone" name="contactPhone" required>
        </div>
      </div>

      <div class="form-group">
        <label for="email">Email Address</label>
        <input type="email" id="email" name="email" placeholder="xxxxx@ecil.co.in"
required>
      </div>

      <h2 class="section-title">Particulars of the System</h2>

      <div class="form-row">
        <div class="form-group">
          <label for="systemMake">System Make, Model etc.</label>
          <input type="text" id="systemMake" name="systemMake" required>
        </div>
        <div class="form-group">
          <label for="configuration">Configuration (CPU, RAM, HDD etc.)</label>
          <input type="text" id="configuration" name="configuration" required>
        </div>
      </div>

      <div class="form-row">
        <div class="form-group">
```

```
        <label for="osName">Operating System Name & Version</label>
        <input type="text" id="osName" name="osName" placeholder="e.g. Windows 10
64-bit" required>
      </div>
      <div class="form-group">
        <label for="licenseDetails">License Details of the OS</label>
        <input type="text" id="licenseDetails" name="licenseDetails" required>
      </div>
    </div>

    <div class="form-row">
      <div class="form-group">
        <label for="antivirusName">Anti-Virus Name & Version</label>
        <input type="text" id="antivirusName" name="antivirusName" required>
      </div>
      <div class="form-group">
        <label for="macAddress">MAC Address</label>
        <input type="text" id="macAddress" name="macAddress" required>
      </div>
    </div>

    <div class="form-group">
      <label for="ipAddress">IP Address (to be assigned by Admin)</label>
      <input type="text" id="ipAddress" name="ipAddress" placeholder="Leave blank -
will be assigned" disabled>
    </div>

    <div class="checkbox-group">
      <input type="checkbox" id="agreement" name="agreement" required>
      <label for="agreement">
        I understand that the use of Internet is for official purposes only. I certify that only
licensed software is being used on my machine. I would abide by the guidelines and rules
regarding usage of Internet account as pronounced by the management from time to
time.
      </label>
    </div>

    <div class="form-group">
      <label for="applicantSignature">Signature of the Applicant</label>
      <input type="text" id="applicantSignature" name="applicantSignature" readonly>
    </div>

    <div class="form-group">
      <label for="itChampion">Forward to IT Champion</label>
      <select id="itChampion" name="itChampion" required>
        <option value="">Select IT Champion</option>
        <option value="Alice Johnson">champion1</option>
      </select>
    </div>

    <div class="submit-section">
```

```
      <button type="submit" class="submit-btn" id="submitBtn">Submit Request</
button>
    </div>
   </form>
  </div>

  <script>
   const API_BASE = '';

   // Check if user is logged in
   const userInfo = JSON.parse(sessionStorage.getItem('userInfo') || '{}');
   if (!userInfo.username || userInfo.role !== 'NormalUser') {
    window.location.href = 'index.html';
   }

   // Populate user info
   document.getElementById('currentUser').textContent = userInfo.username;
   document.getElementById('applicantName').value = userInfo.username;
   document.getElementById('applicantSignature').value = userInfo.username;

   // Handle form submission
   document.getElementById('requestForm').addEventListener('submit', async function(e)
{
    e.preventDefault();

    const submitBtn = document.getElementById('submitBtn');
    const messageDiv = document.getElementById('message');

    // Show loading state
    submitBtn.disabled = true;
    submitBtn.textContent = 'Submitting...';
    messageDiv.style.display = 'none';

    try {
     // Collect form data
     const formData = new FormData(this);
     const requestData = {
      Department: formData.get('divSecCode') || 'General',
      // Store additional details in a structured way (you might want to add more fields to
your Request model)
      Details: {
       applicantName: formData.get('applicantName'),
       divSecCode: formData.get('divSecCode'),
       designation: formData.get('designation'),
       contactPhone: formData.get('contactPhone'),
       email: formData.get('email'),
       systemMake: formData.get('systemMake'),
       configuration: formData.get('configuration'),
       osName: formData.get('osName'),
       licenseDetails: formData.get('licenseDetails'),
       antivirusName: formData.get('antivirusName'),
       macAddress: formData.get('macAddress'),
```

```
          itChampion: formData.get('itChampion')
        }
      };

      const response = await fetch(`SubmitRequestHandler.ashx`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(requestData)
      });

      if (response.ok) {
        const result = await response.json();
        showMessage('Request submitted successfully! Request ID: ' + result.Id,
'success');
        this.reset();
        // Repopulate readonly fields
        document.getElementById('applicantName').value = userInfo.username;
        document.getElementById('applicantSignature').value = userInfo.username;
      } else {
        const error = await response.json();
        throw new Error(error.message || 'Failed to submit request');
      }
    } catch (error) {
      showMessage('Error: ' + error.message, 'error');
    } finally {
      submitBtn.disabled = false;
      submitBtn.textContent = 'Submit Request';
    }
  });

  function showMessage(text, type) {
    const messageDiv = document.getElementById('message');
    messageDiv.textContent = text;
    messageDiv.className = `message ${type}`;
    messageDiv.style.display = 'block';

    // Auto-hide success messages after 5 seconds
    if (type === 'success') {
      setTimeout(() => {
        messageDiv.style.display = 'none';
      }, 5000);
    }
  }

  function logout() {
    sessionStorage.removeItem('userInfo');
    window.location.href = 'index.html';
  }
 </script>
</body>
```

```
</html>
```

itChampion.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>IT Champion - INTERNET Requests</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 20px;
            background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
            min-height: 100vh;
        }

        .header {
            background: white;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 4px 10px rgba(0,0,0,0.1);
            margin-bottom: 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }

        .header h1 {
            color: #333;
            margin: 0;
            font-size: 24px;
        }

        .user-info {
            color: #666;
            font-size: 14px;
        }

        .logout-btn {
            background: #dc3545;
            color: white;
            padding: 8px 16px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            text-decoration: none;
            font-size: 14px;
        }
```

```
.stats-container {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    margin-bottom: 20px;
    padding: 20px;
}

.stats-title {
    color: #333;
    font-size: 18px;
    font-weight: 600;
    margin-bottom: 15px;
    text-align: center;
}

.stats-table {
    width: 100%;
    border-collapse: collapse;
    max-width: 600px;
    margin: 0 auto;
}

.stats-table th,
.stats-table td {
    padding: 12px 20px;
    text-align: center;
    border: 1px solid #e9ecef;
}

.stats-table th {
    background: linear-gradient(135deg, #2196F3 0%, #1976D2 100%);
    color: white;
    font-weight: 600;
    font-size: 14px;
}

.stats-table td {
    background: #f8f9fa;
    font-size: 16px;
    font-weight: 600;
    color: #495057;
}

.stats-count {
    font-size: 24px;
    color: #2196F3;
}

.requests-container {
    background: white;
```

```
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    overflow: hidden;
    margin-bottom: 20px;
}

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

th, td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #eee;
}

th {
    background: linear-gradient(135deg, #2196F3 0%, #1976D2 100%);
    color: white;
    font-weight: 600;
}

tr:hover {
    background-color: #f8f9fa;
    cursor: pointer;
}

tr.selected {
    background-color: #e3f2fd !important;
}

.status-badge {
    padding: 4px 8px;
    border-radius: 12px;
    font-size: 12px;
    font-weight: 600;
    text-transform: uppercase;
}

.status-pending {
    background: #fff3cd;
    color: #856404;
}

.status-hod-approved {
    background: #d1ecf1;
    color: #0c5460;
}

.checklist-panel {
    background: white;
```

```
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    padding: 30px;
    display: none;
}

.checklist-header {
    text-align: center;
    margin-bottom: 30px;
    padding-bottom: 20px;
    border-bottom: 2px solid #eee;
}

.checklist-header h2 {
    margin: 0;
    color: #333;
    font-size: 22px;
}

.ecnet-details {
    background: #f8f9fa;
    padding: 20px;
    border-radius: 8px;
    margin-bottom: 20px;
}

.ecnet-table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

.ecnet-table th, .ecnet-table td {
    border: 1px solid #333;
    padding: 12px 15px;
    text-align: left;
    vertical-align: middle;
}

.ecnet-table th {
    background: #e9ecef;
    font-weight: bold;
    font-size: 14px;
    color: #495057;
    width: 40%;
    white-space: nowrap;
}

.ecnet-table td {
    background: white;
}
```

```
.ecnet-table input {
    width: 100%;
    border: 1px solid #ced4da;
    background: white;
    padding: 8px 12px;
    border-radius: 4px;
    font-size: 14px;
    box-sizing: border-box;
}

.ecnet-table input:focus {
    outline: none;
    border-color: #2196F3;
    box-shadow: 0 0 0 2px rgba(33, 150, 243, 0.2);
}

.section-title {
    font-weight: bold;
    margin: 20px 0 15px 0;
    color: #333;
    font-size: 16px;
    border-bottom: 2px solid #2196F3;
    padding-bottom: 5px;
}

.checkbox-item {
    display: flex;
    align-items: flex-start;
    margin-bottom: 15px;
    padding: 10px;
    border-radius: 6px;
    transition: background-color 0.2s;
    border: 1px solid #e9ecef;
}

.checkbox-item:hover {
    background-color: #f8f9fa;
}

.checkbox-item input[type="checkbox"] {
    margin-right: 12px;
    margin-top: 3px;
    transform: scale(1.2);
}

.checkbox-item label {
    font-size: 14px;
    line-height: 1.4;
    cursor: pointer;
    flex: 1;
}
```

```
.antivirus-details {
    display: none;
    margin-top: 10px;
    padding: 10px;
    background: #e3f2fd;
    border-radius: 4px;
    border-left: 4px solid #2196F3;
}

.antivirus-details.show {
    display: block;
}

.antivirus-row {
    display: flex;
    gap: 15px;
    margin-top: 8px;
}

.antivirus-field {
    flex: 1;
}

.antivirus-field label {
    display: block;
    margin-bottom: 5px;
    font-weight: 500;
    color: #333;
    font-size: 12px;
}

.antivirus-field input {
    width: 100%;
    padding: 6px 8px;
    border: 1px solid #ced4da;
    border-radius: 4px;
    font-size: 13px;
    box-sizing: border-box;
}

.champion-details {
    background: #e8f4fd;
    padding: 20px;
    border-radius: 8px;
    margin: 20px 0;
    border-left: 4px solid #2196F3;
}

.champion-details h3 {
    margin-top: 0;
    color: #1976D2;
    font-size: 18px;
```

```
}

.readonly-note {
    font-style: italic;
    color: #666;
    font-size: 13px;
    margin-bottom: 15px;
}

.form-row {
    display: flex;
    gap: 15px;
    margin-bottom: 15px;
}

.form-group {
    flex: 1;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-weight: 500;
    color: #333;
}

.form-group input, .form-group select, .form-group textarea {
    width: 100%;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 14px;
    box-sizing: border-box;
}

.form-group input:read-only {
    background-color: #f8f9fa;
    color: #6c757d;
    cursor: not-allowed;
}

.form-group textarea {
    min-height: 80px;
    resize: vertical;
}

.action-buttons {
    text-align: center;
    margin-top: 30px;
    padding-top: 20px;
    border-top: 1px solid #eee;
}
```

```
.btn {
    padding: 12px 30px;
    border: none;
    border-radius: 6px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    margin: 0 10px;
    transition: all 0.3s ease;
}

.btn-submit {
    background: linear-gradient(135deg, #28a745 0%, #20c997 100%);
    color: white;
}

.btn-submit:hover {
    transform: translateY(-1px);
    box-shadow: 0 4px 8px rgba(40, 167, 69, 0.3);
}

.btn-cancel {
    background: #6c757d;
    color: white;
}

.loading {
    text-align: center;
    padding: 40px;
    color: #666;
}

.message {
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 20px;
    display: none;
}

.success {
    background: #d4edda;
    color: #155724;
    border: 1px solid #c3e6cb;
}

.error {
    background: #f8d7da;
    color: #721c24;
    border: 1px solid #f5c6cb;
}
```

```
        @media (max-width: 768px) {
            .form-row, .antivirus-row {
                flex-direction: column;
            }

            .ecnet-table {
                font-size: 12px;
            }

            .ecnet-table th {
                font-size: 12px;
            }

            .action-buttons .btn {
                display: block;
                margin-bottom: 10px;
            }

            .stats-table th,
            .stats-table td {
                padding: 8px 12px;
                font-size: 14px;
            }

            .stats-count {
                font-size: 20px;
            }
        }
    </style>
</head>
<body>
    <div class="header">
        <div>
            <h1>IT Champion Dashboard</h1>
            <div class="user-info">
                <strong>Logged in as:</strong> <span id="currentUser">IT Champion</span>
(IT Champion)
            </div>
        </div>
        <button class="logout-btn" onclick="logout()">Logout</button>
    </div>

    <div id="message" class="message"></div>

    <div class="stats-container">
        <div class="stats-title">Request Statistics</div>
        <table class="stats-table">
            <thead>
                <tr>
                    <th>Total Forwarded to HOD</th>
                    <th>HOD Approved</th>
                    </tr>
```

```
            </thead>
            <tbody>
                <tr>
                    <td><span id="totalForwarded" class="stats-count">0</span></td>
                    <td><span id="hodApproved" class="stats-count">0</span></td>
                    </tr>
            </tbody>
        </table>
    </div>

    <div class="requests-container">
        <div id="loadingMessage" class="loading">Loading requests...</div>
        <table id="requestsTable" style="display: none;">
            <thead>
                <tr>
                    <th>Request ID</th>
                    <th>Applicant Name</th>
                    <th>Department</th>
                    <th>System Make</th>
                    <th>Request Date</th>
                    <th>Status</th>
                    <th>Workflow Stage</th>
                </tr>
            </thead>
            <tbody id="requestsTableBody">
            </tbody>
        </table>
    </div>

    <div id="checklistPanel" class="checklist-panel">
        <div class="checklist-header">
            <h2>Checklist for Desktop Security Compliance</h2>
        </div>

        <div class="ecnet-details">
            <h3>Particulars of the ECNET Connectivity Point</h3>
            <p><strong>Please fill in the connectivity details below:</strong></p>

            <table class="ecnet-table">
                <tr>
                    <th>Building Switch IP Address</th>
                    <td><input type="text" id="buildingSwitchIP" placeholder="Enter Building
Switch IP Address"></td>
                </tr>
                <tr>
                    <th>Building Switch Port Number</th>
                    <td><input type="text" id="buildingSwitchPort" placeholder="Enter Building
Switch Port Number"></td>
                </tr>
                <tr>
                    <th>Access Switch IP Address</th>
```

```
            <td><input type="text" id="accessSwitchIP" placeholder="Enter Access
Switch IP Address"></td>
          </tr>
          <tr>
            <th>Access Switch Port Number</th>
            <td><input type="text" id="accessSwitchPort" placeholder="Enter Access
Switch Port Number"></td>
          </tr>
          <tr>
            <th>I/O Port Number</th>
            <td><input type="text" id="ioPortNo" placeholder="Enter I/O Port
Number"></td>
          </tr>
        </table>
    </div>

    <div class="section-title">Configuration Parameters:</div>

    <div class="checkbox-item">
        <input type="checkbox" id="legalOS">
        <label for="legalOS">Legal OS and Application Software</label>
    </div>

    <div class="checkbox-item">
        <input type="checkbox" id="antivirus" onchange="toggleAntivirusDetails()">
        <label for="antivirus">Antivirus Software Installed</label>
        <div id="antivirusDetails" class="antivirus-details">
            <div class="antivirus-row">
                <div class="antivirus-field">
                    <label for="antivirusBrand">Antivirus Brand:</label>
                    <input type="text" id="antivirusBrand" placeholder="e.g., McAfee, Norton,
Kaspersky">
                </div>
                <div class="antivirus-field">
                    <label for="antivirusVersion">Version:</label>
                    <input type="text" id="antivirusVersion" placeholder="e.g., 2024.1.0">
                </div>
            </div>
        </div>
    </div>

    <div class="checkbox-item">
        <input type="checkbox" id="usbControl">
        <label for="usbControl">Installed USB Pratirodh (Controlled removable media
usage)</label>
    </div>

    <div class="checkbox-item">
        <input type="checkbox" id="strongPassword">
        <label for="strongPassword">Strong Login Password (Minimum 8 characters with
combination of alphabets, numerals, and special characters)</label>
    </div>
```

```
        <div class="checkbox-item">
            <input type="checkbox" id="separateAccounts">
            <label for="separateAccounts">Separate Administrator & User accounts
configured</label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="disableRDP">
            <label for="disableRDP">Remote Desktop features disabled</label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="disableAutorun">
            <label for="disableAutorun">Auto-run/Auto-play features disabled</label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="windowsFirewall">
            <label for="windowsFirewall">Windows Firewall enabled</label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="fileSharing">
            <label for="fileSharing">File & Printer sharing turned off</label>
        </div>

        <div class="section-title">Operational Practices:</div>

        <div class="checkbox-item">
            <input type="checkbox" id="securityUpdates">
            <label for="securityUpdates">Periodic check of security updates & patches by IT
Champion</label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="nonAdminUsage">
            <label for="nonAdminUsage">Non-use of Administrator account for daily work</
label>
        </div>

        <div class="checkbox-item">
            <input type="checkbox" id="emailBackup">
            <label for="emailBackup">Weekly desktop mail backup using Outlook ensured</
label>
        </div>

        <div class="champion-details">
            <h3>IT Champion Details</h3>
            <div class="readonly-note">Note: Champion details are automatically fetched
from the database and cannot be modified.</div>
            <div class="form-row">
```

```
                <div class="form-group">
                    <label for="championName">Champion Name:</label>
                    <input type="text" id="championName" readonly>
                </div>
                <div class="form-group">
                    <label for="championCode">Champion Code:</label>
                    <input type="text" id="championCode" readonly>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group">
                    <label for="championDepartment">Department:</label>
                    <input type="text" id="championDepartment" readonly>
                </div>
                <div class="form-group">
                    <label for="championPhone">Phone Number:</label>
                    <input type="tel" id="championPhone" readonly>
                </div>
            </div>
            <div class="form-row">
                <div class="form-group">
                    <label for="championEmail">Email Address:</label>
                    <input type="email" id="championEmail" readonly>
                </div>
                <div class="form-group">
                    <label for="verificationDate">Verification Date:</label>
                    <input type="date" id="verificationDate" readonly>
                </div>
            </div>
            <div class="form-group">
                <label for="itChampionComments">IT Champion Review Comments:</label>
                <textarea id="itChampionComments" placeholder="Enter your review
comments here..."></textarea>
            </div>
        </div>

        <div class="action-buttons">
            <button class="btn btn-submit" onclick="forwardToHOD()">Forward to HOD</
button>
            <button class="btn btn-cancel" onclick="cancelReview()">Cancel</button>
        </div>
    </div>

<script>
    // Configuration - Update this to match your backend handler URL
    const HANDLER_URL = 'ViewRequestHandler.ashx'; // Update this path as needed

    // Global variables
    let currentRequests = [];
    let allRequests = []; // Store all requests for statistics
    let selectedRequest = null;
    let championData = {
```

```
        name: 'IT Champion',
        code: 'IT001',
        department: 'IT Department',
        phone: '1234567890',
        email: 'itchampion@company.com'
    };

    // Initialize page
    document.addEventListener('DOMContentLoaded', function () {
        document.getElementById('verificationDate').value = new
Date().toISOString().split('T')[0];
        loadChampionDetails();
        loadRequests();
    });

    async function loadRequests() {
        try {
            const response = await fetch(`${HANDLER_URL}?role=it-champion`);
            if (!response.ok) throw new Error('Failed to fetch requests');

            const result = await response.json();
            console.log('Raw API Response:', result);

            allRequests = result.Requests || []; // Store all requests
            currentRequests = allRequests;
            console.log('Total requests received:', currentRequests.length);

            // Filter requests relevant for IT Champion review
            const pendingRequests = currentRequests.filter(req => {
                const isForITChampion =
                    (req.Status === 'Pending' && req.WorkflowStage === 'Submitted') ||
                    req.WorkflowStage === 'IT Champion Review' ||
                    req.WorkflowStage === 'Pending IT Champion Review' ||
                    req.WorkflowStage === 'IT Review';
                return isForITChampion;
            });

            console.log('Filtered requests for IT Champion:', pendingRequests.length);

            displayRequests(pendingRequests);
            updateRequestStatistics(); // Update statistics after loading requests

            document.getElementById('loadingMessage').style.display = 'none';
            document.getElementById('requestsTable').style.display = 'table';
        } catch (error) {
            console.error('Error in loadRequests:', error);
            showMessage('Error loading requests: ' + error.message, 'error');
            document.getElementById('loadingMessage').innerHTML = 'Error loading
requests: ' + error.message;
        }
    }
```

```
    function updateRequestStatistics() {
        // Count requests forwarded to HOD (requests that have passed IT Champion
review)
        const forwardedRequests = allRequests.filter(req => {
            return req.WorkflowStage === 'HOD Review' ||
                   req.WorkflowStage === 'HOD Approved' ||
                   req.WorkflowStage === 'Approved' ||
                   req.WorkflowStage === 'Completed' ||
                   (req.Status && req.Status.includes('HOD'));
        });

        // Count HOD approved requests
        const hodApprovedRequests = allRequests.filter(req => {
            return req.WorkflowStage === 'HOD Approved' ||
                   req.WorkflowStage === 'Approved' ||
                   req.WorkflowStage === 'Completed' ||
                   (req.Status && (req.Status.includes('Approved') || req.Status === 'HOD
Approved'));
        });

        const totalForwarded = forwardedRequests.length;
        const totalApproved = hodApprovedRequests.length;
        // Approval rate is no longer calculated or displayed

        // Update the display
        document.getElementById('totalForwarded').textContent = totalForwarded;
        document.getElementById('hodApproved').textContent = totalApproved;

        console.log('Statistics updated:', {
            totalForwarded,
            totalApproved
        });
    }

    function displayRequests(requests) {
        const tableBody = document.getElementById('requestsTableBody');
        tableBody.innerHTML = '';

        if (requests.length === 0) {
            const row = document.createElement('tr');
            const cell = document.createElement('td');
            cell.colSpan = 7;
            cell.textContent = 'No requests available for review.';
            cell.style.textAlign = 'center';
            row.appendChild(cell);
            tableBody.appendChild(row);
            return;
        }

        requests.forEach(request => {
            const row = document.createElement('tr');
            row.addEventListener('click', () => selectRequest(request, row));
```

```
            row.innerHTML = `
                <td>${request.RequestID || request.Id}</td>
                <td>${request.ApplicantName || 'N/A'}</td>
                <td>${request.Department || 'N/A'}</td>
                <td>${request.SystemMake || 'N/A'}</td>
                <td>${formatDate(request.RequestDate)}</td>
                <td><span class="status-badge status-${request.Status.toLowerCase().replace(/\s/g, '-')}">${request.Status}</span></td>
                <td>${request.WorkflowStage}</td>
            `;
            tableBody.appendChild(row);
        });
    }

    function selectRequest(request, row) {
        const previouslySelected = document.querySelector('tr.selected');
        if (previouslySelected) {
            previouslySelected.classList.remove('selected');
        }

        row.classList.add('selected');
        selectedRequest = request;

        // Populate checklist form with request details if available or clear
        populateChecklist(request);

        document.getElementById('checklistPanel').style.display = 'block';
        document.getElementById('checklistPanel').scrollIntoView({ behavior: 'smooth' });
    }

    function populateChecklist(request) {
        // Clear previous data
        resetChecklistForm();

        // Populate ECNET Connectivity Point details if available in the request
        if (request.ConnectivityDetails) {
            document.getElementById('buildingSwitchIP').value = request.ConnectivityDetails.BuildingSwitchIP || '';
            document.getElementById('buildingSwitchPort').value = request.ConnectivityDetails.BuildingSwitchPort || '';
            document.getElementById('accessSwitchIP').value = request.ConnectivityDetails.AccessSwitchIP || '';
            document.getElementById('accessSwitchPort').value = request.ConnectivityDetails.AccessSwitchPort || '';
            document.getElementById('ioPortNo').value = request.ConnectivityDetails.IOPortNo || '';
        }

        // Populate Configuration Parameters if available
        if (request.ConfigChecks) {
```

```
            document.getElementById('legalOS').checked = request.ConfigChecks.LegalOS ||
false;
            document.getElementById('antivirus').checked = request.ConfigChecks.Antivirus ||
false;
            toggleAntivirusDetails(); // Ensure antivirus details panel is shown/hidden correctly
            if (request.ConfigChecks.Antivirus && request.AntivirusDetails) {
                document.getElementById('antivirusBrand').value =
request.AntivirusDetails.Brand || '';
                document.getElementById('antivirusVersion').value =
request.AntivirusDetails.Version || '';
            }
            document.getElementById('usbControl').checked =
request.ConfigChecks.UsbControl || false;
            document.getElementById('strongPassword').checked =
request.ConfigChecks.StrongPassword || false;
            document.getElementById('separateAccounts').checked =
request.ConfigChecks.SeparateAccounts || false;
            document.getElementById('disableRDP').checked =
request.ConfigChecks.DisableRDP || false;
            document.getElementById('disableAutorun').checked =
request.ConfigChecks.DisableAutorun || false;
            document.getElementById('windowsFirewall').checked =
request.ConfigChecks.WindowsFirewall || false;
            document.getElementById('fileSharing').checked =
request.ConfigChecks.FileSharing || false;
        }

        // Populate Operational Practices if available
        if (request.OperationalPractices) {
            document.getElementById('securityUpdates').checked =
request.OperationalPractices.SecurityUpdates || false;
            document.getElementById('nonAdminUsage').checked =
request.OperationalPractices.NonAdminUsage || false;
            document.getElementById('emailBackup').checked =
request.OperationalPractices.EmailBackup || false;
        }

        // Populate IT Champion Review Comments if available
        document.getElementById('itChampionComments').value =
request.ITChampionComments || '';
    }

    function loadChampionDetails() {
        document.getElementById('championName').value = championData.name;
        document.getElementById('championCode').value = championData.code;
        document.getElementById('championDepartment').value =
championData.department;
        document.getElementById('championPhone').value = championData.phone;
        document.getElementById('championEmail').value = championData.email;
    }

    function toggleAntivirusDetails() {
```

```
        const checkbox = document.getElementById('antivirus');
        const details = document.getElementById('antivirusDetails');

        if (checkbox.checked) {
            details.classList.add('show');
        } else {
            details.classList.remove('show');
            document.getElementById('antivirusBrand').value = '';
            document.getElementById('antivirusVersion').value = '';
        }
    }

    async function forwardToHOD() {
        if (!selectedRequest) {
            showMessage('Please select a request first', 'error');
            return;
        }

        const antivirusChecked = document.getElementById('antivirus').checked;
        if (antivirusChecked) {
            const brand = document.getElementById('antivirusBrand').value.trim();
            const version = document.getElementById('antivirusVersion').value.trim();
            if (!brand || !version) {
                showMessage('Please provide both antivirus brand and version when antivirus
is checked', 'error');
                return;
            }
        }

        // Collect all checklist data
        const formData = {
            requestId: selectedRequest.RequestID || selectedRequest.Id,
            action: 'forward_to_hod',
            connectivityDetails: {
                buildingSwitchIP: document.getElementById('buildingSwitchIP').value,
                buildingSwitchPort: document.getElementById('buildingSwitchPort').value,
                accessSwitchIP: document.getElementById('accessSwitchIP').value,
                accessSwitchPort: document.getElementById('accessSwitchPort').value,
                ioPortNo: document.getElementById('ioPortNo').value
            },
            configChecks: {
                legalOS: document.getElementById('legalOS').checked,
                antivirus: document.getElementById('antivirus').checked,
                usbControl: document.getElementById('usbControl').checked,
                strongPassword: document.getElementById('strongPassword').checked,
                separateAccounts: document.getElementById('separateAccounts').checked,
                disableRDP: document.getElementById('disableRDP').checked,
                disableAutorun: document.getElementById('disableAutorun').checked,
                windowsFirewall: document.getElementById('windowsFirewall').checked,
                fileSharing: document.getElementById('fileSharing').checked,
            },
            operationalPractices: { // Added this new section for operational practices
```

```
            securityUpdates: document.getElementById('securityUpdates').checked,
            nonAdminUsage: document.getElementById('nonAdminUsage').checked,
            emailBackup: document.getElementById('emailBackup').checked
        },
        antivirusDetails: antivirusChecked ? {
            brand: document.getElementById('antivirusBrand').value,
            version: document.getElementById('antivirusVersion').value
        } : null,
        championDetails: championData,
        verificationDate: document.getElementById('verificationDate').value,
        comments: document.getElementById('itChampionComments').value
    };

    console.log('Submitting form data:', formData);

    try {
        const response = await fetch(HANDLER_URL, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(formData)
        });

        console.log('Response status:', response.status);

        if (!response.ok) {
            const errorText = await response.text();
            console.error('Response error:', errorText);
            throw new Error(`HTTP ${response.status}: ${errorText}`);
        }

        const result = await response.json();
        console.log('Response result:', result);

        if (result.success) {
            showMessage('Request successfully forwarded to HOD for approval!',
'success');
            resetChecklistForm();
            document.getElementById('checklistPanel').style.display = 'none'; // Hide
checklist after submission
            selectedRequest = null; // Clear selected request
            // Remove the selected class from the table row
            const previouslySelected = document.querySelector('tr.selected');
            if (previouslySelected) {
                previouslySelected.classList.remove('selected');
            }
            setTimeout(() => {
                loadRequests(); // Reload requests to update the table and statistics
            }, 1000); // Give a small delay for the message to be seen
        } else {
            showMessage(result.message || 'Failed to forward request to HOD.', 'error');
```

```
            }
        } catch (error) {
            console.error('Error forwarding request:', error);
            showMessage('Error forwarding request: ' + error.message, 'error');
        }
    }

    function resetChecklistForm() {
        // Clear ECNET connectivity details
        document.getElementById('buildingSwitchIP').value = '';
        document.getElementById('buildingSwitchPort').value = '';
        document.getElementById('accessSwitchIP').value = '';
        document.getElementById('accessSwitchPort').value = '';
        document.getElementById('ioPortNo').value = '';

        // Uncheck all configuration and operational practice checkboxes
        document.getElementById('legalOS').checked = false;
        document.getElementById('antivirus').checked = false;
        toggleAntivirusDetails(); // Hide antivirus details and clear inputs
        document.getElementById('usbControl').checked = false;
        document.getElementById('strongPassword').checked = false;
        document.getElementById('separateAccounts').checked = false;
        document.getElementById('disableRDP').checked = false;
        document.getElementById('disableAutorun').checked = false;
        document.getElementById('windowsFirewall').checked = false;
        document.getElementById('fileSharing').checked = false;
        document.getElementById('securityUpdates').checked = false;
        document.getElementById('nonAdminUsage').checked = false;
        document.getElementById('emailBackup').checked = false;

        // Clear IT Champion comments
        document.getElementById('itChampionComments').value = '';

        // Reset verification date
        document.getElementById('verificationDate').value = new
Date().toISOString().split('T')[0];
    }

    function cancelReview() {
        resetChecklistForm();
        document.getElementById('checklistPanel').style.display = 'none';
        selectedRequest = null; // Clear selected request
        const previouslySelected = document.querySelector('tr.selected');
        if (previouslySelected) {
            previouslySelected.classList.remove('selected');
        }
        showMessage('Review cancelled.', 'info'); // Added an info message for cancel
    }

    function showMessage(msg, type) {
        const messageDiv = document.getElementById('message');
        messageDiv.textContent = msg;
```

```
        messageDiv.className = `message ${type}`; // Apply 'message' class and specific
type (success/error)
        messageDiv.style.display = 'block';
        setTimeout(() => {
            messageDiv.style.display = 'none';
            messageDiv.textContent = '';
        }, 5000); // Hide message after 5 seconds
    }

    function formatDate(dateString) {
        if (!dateString) return 'N/A';
        const date = new Date(dateString);
        if (isNaN(date)) return 'Invalid Date';
        return date.toLocaleDateString('en-CA'); // YYYY-MM-DD format
    }

    function logout() {
        // In a real application, this would redirect to a login page or clear session.
        alert('Logging out...');
        window.location.href = '/logout'; // Example redirect
    }
</script>
</body>
</html>
```

hod.html

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HOD INTERNET Requests</title>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 20px;
            background: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
            min-height: 100vh;
        }

        .header {
            background: white;
            padding: 20px;
            border-radius: 15px;
            box-shadow: 0 5px 15px rgba(0,0,0,0.1);
            margin-bottom: 30px;
            display: flex;
            justify-content: space-between;
            align-items: center;
```

```
}

.header h1 {
    color: #333;
    margin: 0;
    font-size: 28px;
}

.user-info {
    color: #666;
    font-size: 14px;
}

.logout-btn {
    background: #dc3545;
    color: white;
    padding: 8px 16px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-decoration: none;
    font-size: 14px;
}

.stats-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.stat-card {
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    text-align: center;
}

.stat-number {
    font-size: 2em;
    font-weight: bold;
    color: #2196F3;
    margin-bottom: 5px;
}

.stat-label {
    color: #666;
    font-size: 14px;
}

.filter-section {
```

```
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    margin-bottom: 20px;
}

.filter-controls {
    display: flex;
    flex-wrap: wrap;
    gap: 15px;
    align-items: center;
}

.filter-controls select, .filter-controls input {
    padding: 8px 12px;
    border: 2px solid #e1e1e1;
    border-radius: 5px;
    font-size: 14px;
}

.filter-controls select:focus, .filter-controls input:focus {
    outline: none;
    border-color: #2196F3;
}

.table-container {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    overflow: hidden;
    margin-bottom: 20px;
}

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

th, td {
    padding: 15px;
    text-align: left;
    border-bottom: 1px solid #eee;
}

th {
    background: linear-gradient(135deg, #2196F3 0%, #1976D2 100%);
    color: white;
    font-weight: 600;
}

tr:hover {
```

```
    background-color: #f8f9fa;
    cursor: pointer;
}

tr.selected {
    background-color: #e3f2fd !important;
}

.status-badge {
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 12px;
    font-weight: 600;
    text-transform: uppercase;
}

.status-pending {
    background: #fff3cd;
    color: #856404;
}

.status-approved {
    background: #d4edda;
    color: #155724;
}

.status-declined {
    background: #f8d7da;
    color: #721c24;
}

.details-panel {
    background: white;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
    padding: 30px;
    display: none;
}

.details-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.detail-section {
    background: #f8f9fa;
    padding: 20px;
    border-radius: 8px;
}
```

```
.detail-section h3 {
    margin-top: 0;
    color: #333;
    border-bottom: 2px solid #2196F3;
    padding-bottom: 10px;
}

.detail-item {
    margin-bottom: 10px;
}

.detail-label {
    font-weight: 600;
    color: #555;
    display: inline-block;
    width: 150px;
}

.detail-value {
    color: #333;
}

.action-buttons {
    display: flex;
    gap: 15px;
    justify-content: center;
    margin-top: 30px;
    padding-top: 20px;
    border-top: 1px solid #eee;
}

.btn {
    padding: 12px 30px;
    border: none;
    border-radius: 8px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.3s ease;
}

.btn-approve {
    background: linear-gradient(135deg, #28a745 0%, #20c997 100%);
    color: white;
}

.btn-approve:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(40, 167, 69, 0.3);
}

.btn-decline {
```

```
    background: linear-gradient(135deg, #dc3545 0%, #c82333 100%);
    color: white;
}

.btn-decline:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(220, 53, 69, 0.3);
}

.btn:disabled {
    background: #6c757d;
    cursor: not-allowed;
    transform: none;
}

.loading {
    text-align: center;
    padding: 40px;
    color: #666;
}

.error {
    background: #f8d7da;
    color: #721c24;
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 20px;
}

.comments-section {
    margin-top: 20px;
}

.comments-section textarea {
    width: 100%;
    min-height: 100px;
    padding: 12px;
    border: 2px solid #e1e1e1;
    border-radius: 8px;
    font-family: inherit;
    resize: vertical;
}

@media (max-width: 768px) {
    .filter-controls {
        flex-direction: column;
        align-items: stretch;
    }

    .details-grid {
        grid-template-columns: 1fr;
    }
```

```
        .action-buttons {
            flex-direction: column;
        }
    }
</style>
</head>
<body>
<div class="header">
    <div>
        <h1>INTERNET Request Management</h1>
        <div class="user-info">
            <strong>Logged in as:</strong> <span id="currentUser"></span> (HOD)
        </div>
    </div>
    <button class="logout-btn" onclick="logout()">Logout</button>
</div>

<div class="stats-container">
    <div class="stat-card">
        <div class="stat-number" id="totalRequests">0</div>
        <div class="stat-label">Total Requests</div>
    </div>
    <div class="stat-card">
        <div class="stat-number" id="pendingRequests">0</div>
        <div class="stat-label">Pending Approval</div>
    </div>
    <div class="stat-card">
        <div class="stat-number" id="approvedRequests">0</div>
        <div class="stat-label">Approved</div>
    </div>
    <div class="stat-card">
        <div class="stat-number" id="declinedRequests">0</div>
        <div class="stat-label">Declined</div>
    </div>
</div>

<div id="errorMessage" class="error" style="display: none;"></div>

<div class="filter-section">
    <div class="filter-controls">
    <div class="filter-controls">
        <label>
            Department:
            <select id="deptFilter">
                <option value="">All Departments</option>
            </select>
        </label>
        <label>
            Status:
            <select id="statusFilter">
                <option value="">All Status</option>
```

```
                <option value="Pending">Pending</option>
                <option value="HOD Approved">HOD Approved</option>
                <option value="HOD Rejected">HOD Rejected</option>
                <option value="Completed">Completed</option>
            </select>
        </label>
        <label>
            From Date:
            <input type="date" id="fromDate">
        </label>
        <label>
            To Date:
            <input type="date" id="toDate">
        </label>
        <button class="btn" onclick="applyFilters()" style="background: #2196F3; color:
white;">Apply Filters</button>
    </div>
</div>

<div class="table-container">
    <div id="loadingMessage" class="loading">Loading requests...</div>
    <table id="requestsTable" style="display: none;">
        <thead>
            <tr>
                <th>Request ID</th>
                <th>Applicant</th>
                <th>Department</th>
                <th>Request Date</th>
                <th>Status</th>
                <th>IT Champion</th>
            </tr>
        </thead>
        <tbody id="requestsTableBody">
        </tbody>
    </table>
</div>

<div id="detailsPanel" class="details-panel">
    <h2>Request Details</h2>

    <div class="details-grid">
        <div class="detail-section">
            <h3>Applicant Information</h3>
            <div class="detail-item">
                <span class="detail-label">Name:</span>
                <span class="detail-value" id="detail-applicantName"></span>
            </div>
            <div class="detail-item">
                <span class="detail-label">Department:</span>
                <span class="detail-value" id="detail-department"></span>
            </div>
            <div class="detail-item">
```

```
        <span class="detail-label">Designation:</span>
        <span class="detail-value" id="detail-designation"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Email:</span>
        <span class="detail-value" id="detail-email"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Phone:</span>
        <span class="detail-value" id="detail-phone"></span>
    </div>
</div>

<div class="detail-section">
    <h3>System Information</h3>
    <div class="detail-item">
        <span class="detail-label">System Make:</span>
        <span class="detail-value" id="detail-systemMake"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Configuration:</span>
        <span class="detail-value" id="detail-configuration"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">OS:</span>
        <span class="detail-value" id="detail-os"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Antivirus:</span>
        <span class="detail-value" id="detail-antivirus"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">MAC Address:</span>
        <span class="detail-value" id="detail-mac"></span>
    </div>
</div>

<div class="detail-section">
    <h3>IT Champion Review</h3>
    <div class="detail-item">
        <span class="detail-label">Champion:</span>
        <span class="detail-value" id="detail-champion"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Review Date:</span>
        <span class="detail-value" id="detail-reviewDate"></span>
    </div>
    <div class="detail-item">
        <span class="detail-label">Workflow Stage:</span>
        <span class="detail-value" id="detail-checklistStatus"></span>
    </div>
</div>
```

```
        </div>

        <div class="comments-section">
            <h3>HOD Comments</h3>
            <textarea id="hodComments" placeholder="Add your comments here..."></
textarea>
        </div>

        <div class="action-buttons">
            <button class="btn btn-approve" id="approveBtn"
onclick="updateRequestStatus('Approved')">
                Approve Request
            </button>
            <button class="btn btn-decline" id="declineBtn"
onclick="updateRequestStatus('Declined')">
                Decline Request
            </button>
        </div>
    </div>

    <script>
        const API_BASE = '';
        let currentRequests = [];
        let selectedRequest = null;

        // Check authentication
        const userInfo = JSON.parse(sessionStorage.getItem('userInfo') || '{}');
        if (!userInfo.username || userInfo.role !== 'HOD') {
            window.location.href = 'index.html';
        }

        document.getElementById('currentUser').textContent = userInfo.username;

        // Load data on page load
        document.addEventListener('DOMContentLoaded', function() {
            loadRequests();
            // Add event listeners for filter changes
            document.getElementById('deptFilter').addEventListener('change', applyFilters);
            document.getElementById('statusFilter').addEventListener('change', applyFilters);
            document.getElementById('fromDate').addEventListener('change', applyFilters);
            document.getElementById('toDate').addEventListener('change', applyFilters);
        });

        async function loadRequests() {
            try {
                document.getElementById('loadingMessage').style.display = 'block';
                document.getElementById('requestsTable').style.display = 'none';
                document.getElementById('detailsPanel').style.display = 'none';
                showError(''); // Clear any previous error messages

                // Fetch call should go to ViewRequestHandler.ashx with role filter
                const response = await fetch(`ViewRequestHandler.ashx?role=hod`);
```

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

            const responseData = await response.json();

            // Check if response has the expected structure
            if (responseData.Requests && Array.isArray(responseData.Requests)) {
                currentRequests = responseData.Requests;
            } else if (Array.isArray(responseData)) {
                // Fallback: if response is directly an array
                currentRequests = responseData;
            } else {
                throw new Error('Invalid response format: Expected array of requests');
            }

            displayRequests(currentRequests);
            updateStats(currentRequests);
            populateDepartmentFilter(currentRequests);

            document.getElementById('loadingMessage').style.display = 'none';
            document.getElementById('requestsTable').style.display = 'table';
        } catch (error) {
            showError('Error loading requests: ' + error.message);
            document.getElementById('loadingMessage').style.display = 'none';
            console.error('Load requests error:', error);
        }
    }

    function displayRequests(requests) {
        const tbody = document.getElementById('requestsTableBody');
        tbody.innerHTML = '';

        if (requests.length === 0) {
            tbody.innerHTML = '<tr><td colspan="6" style="text-align: center; padding:
20px;">No requests found.</td></tr>';
            return;
        }

        requests.forEach(request => {
            const row = document.createElement('tr');
            row.innerHTML = `
                <td>${request.Id}</td>
                <td>${request.ApplicantName || 'N/A'}</td>
                <td>${request.Department || 'N/A'}</td>
                <td>${new Date(request.RequestDate).toLocaleDateString()}</td>
                <td><span class="status-badge status-$
{request.Status.toLowerCase().replace(/\s+/g, '-')}">${request.Status}</span></td>
                <td>${request.ItChampionName || 'Not Assigned'}</td>
            `;
```

```
            row.addEventListener('click', () => selectRequest(request, row));
            tbody.appendChild(row);
        });
    }

    function selectRequest(request, row) {
        // Remove previous selection
        document.querySelectorAll('tr.selected').forEach(r =>
r.classList.remove('selected'));

        // Select current row
        row.classList.add('selected');
        selectedRequest = request;

        // Populate details panel
        populateDetails(request);
        document.getElementById('detailsPanel').style.display = 'block';
    }

    function populateDetails(request) {
        document.getElementById('detail-applicantName').textContent =
request.ApplicantName || 'N/A';
        document.getElementById('detail-department').textContent = request.Department
|| 'N/A';
        document.getElementById('detail-designation').textContent = request.Designation
|| 'N/A';
        document.getElementById('detail-email').textContent = request.Email || 'N/A';
        document.getElementById('detail-phone').textContent = request.ContactPhone ||
'N/A';
        document.getElementById('detail-systemMake').textContent =
request.SystemMake || 'N/A';
        document.getElementById('detail-configuration').textContent =
request.Configuration || 'N/A';
        document.getElementById('detail-os').textContent = request.OsName || 'N/A';
        document.getElementById('detail-antivirus').textContent = request.AntivirusName
|| 'N/A';
        document.getElementById('detail-mac').textContent = request.MacAddress || 'N/
A';
        document.getElementById('detail-champion').textContent =
request.ItChampionName || 'Not Assigned';
        document.getElementById('detail-reviewDate').textContent =
request.ItChampionReviewDate ?
            new Date(request.ItChampionReviewDate).toLocaleDateString() : 'Pending';
        document.getElementById('detail-checklistStatus').textContent =
request.WorkflowStage || 'N/A';

        document.getElementById('hodComments').value = request.HodComments || '';

        // Disable/enable buttons based on status
        const approveBtn = document.getElementById('approveBtn');
        const declineBtn = document.getElementById('declineBtn');
```

```
        if (request.Status !== 'Pending' && !request.Status.includes('Submitted') && !
request.Status.includes('HOD Review')) {
            approveBtn.disabled = true;
            declineBtn.disabled = true;
            approveBtn.textContent = request.Status.includes('Approved') ? 'Already
Approved' : 'Request Processed';
            declineBtn.textContent = request.Status.includes('Rejected') ? 'Already
Rejected' : 'Request Processed';
        } else {
            approveBtn.disabled = false;
            declineBtn.disabled = false;
            approveBtn.textContent = 'Approve Request';
            declineBtn.textContent = 'Decline Request';
        }
    }

    function updateStats(requests) {
        const total = requests.length;
        const pending = requests.filter(r => r.Status === 'Pending' || r.WorkflowStage ===
'Submitted' || r.WorkflowStage === 'HOD Review').length;
        const approved = requests.filter(r => r.Status.includes('Approved')).length;
        const declined = requests.filter(r => r.Status.includes('Rejected')).length;

        document.getElementById('totalRequests').textContent = total;
        document.getElementById('pendingRequests').textContent = pending;
        document.getElementById('approvedRequests').textContent = approved;
        document.getElementById('declinedRequests').textContent = declined;
    }

    function populateDepartmentFilter(requests) {
        const deptFilter = document.getElementById('deptFilter');
        const departments = [...new Set(requests.map(r => r.Department).filter(Boolean))];
        deptFilter.innerHTML = '<option value="">All Departments</option>';

        departments.sort().forEach(dept => {
            const option = document.createElement('option');
            option.value = dept;
            option.textContent = dept;
            deptFilter.appendChild(option);
        });
    }

    function applyFilters() {
        const deptFilter = document.getElementById('deptFilter').value;
        const statusFilter = document.getElementById('statusFilter').value;
        const fromDateStr = document.getElementById('fromDate').value;
        const toDateStr = document.getElementById('toDate').value;

        const filteredRequests = currentRequests.filter(request => {
            let matches = true;

            if (deptFilter && request.Department !== deptFilter) {
```

```
            matches = false;
        }

        if (statusFilter && request.Status !== statusFilter) {
            matches = false;
        }

        if (fromDateStr) {
            const requestDate = new Date(request.RequestDate);
            const fromDate = new Date(fromDateStr);
            fromDate.setHours(0, 0, 0, 0);
            if (requestDate < fromDate) {
                matches = false;
            }
        }
        if (toDateStr) {
            const requestDate = new Date(request.RequestDate);
            const toDate = new Date(toDateStr);
            toDate.setHours(23, 59, 59, 999);
            if (requestDate > toDate) {
                matches = false;
            }
        }
        return matches;
    });

    displayRequests(filteredRequests);
}

async function updateRequestStatus(newStatus) {
    if (!selectedRequest) {
        showError('No request selected.');
        return;
    }

    const hodComments = document.getElementById('hodComments').value;
    const approveBtn = document.getElementById('approveBtn');
    const declineBtn = document.getElementById('declineBtn');

    approveBtn.disabled = true;
    declineBtn.disabled = true;
    approveBtn.textContent = 'Processing...';
    declineBtn.textContent = 'Processing...';
    showError(''); // Clear previous errors

    try {
        // Determine action based on newStatus
        let action;
        if (newStatus === 'Approved') {
            action = 'hod_approve';
        } else if (newStatus === 'Declined') {
            action = 'hod_reject';
```

```
            } else {
                throw new Error('Invalid status: ' + newStatus);
            }

            // Data to send to the server
            const updateData = {
                Id: selectedRequest.Id,
                Action: action,
                HodComments: hodComments
            };

            // Fetch call should go to ViewRequestHandler.ashx (it handles both GET and
POST)
            const response = await fetch(`ViewRequestHandler.ashx`, {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify(updateData)
            });

            const result = await response.json();

            if (response.ok) {
                alert(result.Message || 'Request updated successfully');
                // Reload requests to reflect changes
                loadRequests();
                // Hide details panel after successful update
                document.getElementById('detailsPanel').style.display = 'none';
            } else {
                throw new Error(result.Message || 'Failed to update request status.');
            }
        } catch (error) {
            showError('Error updating status: ' + error.message);
            console.error('Update request error:', error);
        } finally {
            // Re-enable buttons and restore text after operation
            if (selectedRequest && selectedRequest.Status === 'Pending') {
                approveBtn.disabled = false;
                declineBtn.disabled = false;
                approveBtn.textContent = 'Approve Request';
                declineBtn.textContent = 'Decline Request';
            }
        }
    }

    function showError(message) {
        const errorDiv = document.getElementById('errorMessage');
        if (message) {
            errorDiv.textContent = message;
            errorDiv.style.display = 'block';
        } else {
```

```
            errorDiv.style.display = 'none';
            errorDiv.textContent = '';
        }
    }

    function logout() {
        sessionStorage.removeItem('userInfo');
        window.location.href = 'index.html';
    }
  </script>
</body>
</html>
```

Styles.css

```
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    background: url('images/ecil-logo.png') no-repeat center center fixed;
    background-size: contain;
    background-color: #f4f4f4;
}

.login-box {
    background: rgba(255, 255, 255, 0.85);  /* Mild transparency */
    padding: 25px;
    max-width: 350px;
    margin: 100px auto;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
    backdrop-filter: blur(3px); /* Adds light blur behind form */
}


h2 {
    text-align: center;
}

label {
    margin-top: 12px;
    display: block;
}

input, select {
    width: 100%;
    padding: 10px;
    margin-top: 6px;
    margin-bottom: 12px;
    border-radius: 5px;
    border: 1px solid #ccc;
```

```
    font-size: 14px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #0066cc;
    color: white;
    font-size: 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #004a99;
}

.user-body {
  font-family: Arial, sans-serif;
  background: url('company-logo.jpg') no-repeat center center fixed;
  background-size: cover;
  margin: 0;
  padding: 0;
}

.form-box {
  background: rgba(255, 255, 255, 0.88);
  padding: 30px;
  max-width: 1000px;
  margin: 50px auto;
  border-radius: 12px;
  box-shadow: 0 0 20px rgba(0,0,0,0.3);
  backdrop-filter: blur(3px);
}

.form-box h2 {
  text-align: center;
  margin-bottom: 15px;
}

.form-box label {
  display: block;
  margin-top: 15px;
  font-weight: bold;
}

.form-box input,
.form-box select {
  width: 100%;
  padding: 8px;
  margin-top: 5px;
```

```
  border: 1px solid #aaa;
  border-radius: 5px;
  box-sizing: border-box;
}

.form-box input[readonly] {
  background-color: #f4f4f4;
}

.form-box button {
  margin-top: 20px;
  width: 100%;
  padding: 10px;
  background-color: #005288;
  color: white;
  border: none;
  border-radius: 5px;
  font-weight: bold;
  cursor: pointer;
}
```

LoginHandler.ashx.cs

```
using System;
using System.Web;
using System.Web.Script.Serialization;
using System.Data;
using System.Data.Odbc;
using System.Configuration;

namespace ECNET.Web
{
    public class LoginHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            // Set CORS headers to allow cross-origin requests
            context.Response.AddHeader("Access-Control-Allow-Origin", "*");
            context.Response.AddHeader("Access-Control-Allow-Methods", "GET, POST,
OPTIONS");
            context.Response.AddHeader("Access-Control-Allow-Headers", "Content-Type");

            context.Response.ContentType = "application/json";
            string responseJson = "";
            var jsSerializer = new JavaScriptSerializer();

            // Handle OPTIONS request for CORS preflight
            if (context.Request.HttpMethod == "OPTIONS")
            {
```

```
            context.Response.StatusCode = 200;
            return;
        }

        if (context.Request.HttpMethod == "POST")
        {
            try
            {
                // Read the incoming JSON request body
                string requestBody = new
System.IO.StreamReader(context.Request.InputStream).ReadToEnd();

                // Debug log the request body
                System.Diagnostics.Debug.WriteLine("Request Body: " + requestBody);

                if (string.IsNullOrEmpty(requestBody))
                {
                    context.Response.StatusCode = 400;
                    responseJson = jsSerializer.Serialize(new { Message = "Empty request
body" });
                    context.Response.Write(responseJson);
                    return;
                }

                dynamic loginData = jsSerializer.Deserialize<dynamic>(requestBody);

                // Extract username, password, and role from the deserialized data
                string username = loginData["username"];
                string password = loginData["password"];
                string role = loginData["role"];

                // Validate input
                if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password) ||
string.IsNullOrEmpty(role))
                {
                    context.Response.StatusCode = 400;
                    responseJson = jsSerializer.Serialize(new { Message = "Username,
password, and role are required" });
                    context.Response.Write(responseJson);
                    return;
                }

                // Query the database to authenticate the user
                string query = "SELECT UserId, Username, Role FROM Users WHERE
Username = ? AND Password = ? AND Role = ?";
                OdbcParameter[] parameters = new OdbcParameter[]
                {
                    new OdbcParameter("username", username),
                    new OdbcParameter("password", password),
                    new OdbcParameter("role", role)
                };
```

```
                // Execute the query using DbHelper
                DataTable dt = DbHelper.ExecuteQuery(query, parameters);

                if (dt.Rows.Count > 0)
                {
                    // User found and authenticated
                    var user = new User
                    {
                        UserId = Convert.ToInt32(dt.Rows[0]["UserId"]),
                        Username = dt.Rows[0]["Username"].ToString(),
                        Role = dt.Rows[0]["Role"].ToString()
                    };
                    responseJson = jsSerializer.Serialize(user);
                    context.Response.StatusCode = 200;
                }
                else
                {
                    // Authentication failed
                    context.Response.StatusCode = 401;
                    responseJson = jsSerializer.Serialize(new { Message = "Invalid credentials
or role mismatch." });
                }
            }
            catch (Exception ex)
            {
                // Handle any exceptions during processing
                context.Response.StatusCode = 500;
                responseJson = jsSerializer.Serialize(new { Message = "Server error: " +
ex.Message });

                // Debug log the exception
                System.Diagnostics.Debug.WriteLine("Exception: " + ex.ToString());
            }
        }
        else
        {
            // If not a POST request, return Method Not Allowed
            context.Response.StatusCode = 405;
            responseJson = jsSerializer.Serialize(new { Message = "Only POST requests are
allowed for login." });
        }

        context.Response.Write(responseJson);
    }

    public bool IsReusable
    {
        get { return false; }
    }
  }
}
```

SubmitRequestHandler.ashx.cs

```
using System;
using System.Web;
using System.Web.Script.Serialization;
using System.Data;
using System.Data.Odbc;

namespace ECNET.Web
{
    public class SubmitRequestHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            context.Response.ContentType = "application/json";
            context.Response.AddHeader("Access-Control-Allow-Origin", "*");

            string responseJson = "";
            var jsSerializer = new JavaScriptSerializer();

            if (context.Request.HttpMethod == "POST")
            {
                try
                {
                    string requestBody = new
System.IO.StreamReader(context.Request.InputStream).ReadToEnd();
                    dynamic requestData = jsSerializer.Deserialize<dynamic>(requestBody);

                    dynamic details = requestData["Details"];

                    var newRequest = new Request
                    {
                        ApplicantName = details["applicantName"],
                        Department = details["divSecCode"],
                        Designation = details["designation"],
                        ContactPhone = details["contactPhone"],
                        Email = details["email"],
                        SystemMake = details["systemMake"],
                        Configuration = details["configuration"],
                        OsName = details["osName"],
                        AntivirusName = details["antivirusName"],
                        MacAddress = details["macAddress"],
                        ItChampionName = details["itChampion"],
                        RequestDate = DateTime.Now,
                        Status = "Pending",
                        WorkflowStage = "Submitted",
                        HodComments = null,
                        ItChampionReviewDate = null
                    };

                    // 1. INSERT request (no SELECT here)
```

```
            string insertQuery = @"INSERT INTO Requests (ApplicantName, Department,
Designation, Email, ContactPhone,
                                   SystemMake, Configuration, OsName, AntivirusName,
MacAddress, ItChampionName,
                                   RequestDate, Status, WorkflowStage, HodComments,
ItChampionReviewDate)
                           VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

            OdbcParameter[] parameters = new OdbcParameter[]
            {
                new OdbcParameter("ApplicantName", newRequest.ApplicantName),
                new OdbcParameter("Department", newRequest.Department),
                new OdbcParameter("Designation", newRequest.Designation),
                new OdbcParameter("Email", newRequest.Email),
                new OdbcParameter("ContactPhone", newRequest.ContactPhone),
                new OdbcParameter("SystemMake", newRequest.SystemMake),
                new OdbcParameter("Configuration", newRequest.Configuration),
                new OdbcParameter("OsName", newRequest.OsName),
                new OdbcParameter("AntivirusName",
(object)newRequest.AntivirusName ?? DBNull.Value),
                new OdbcParameter("MacAddress", newRequest.MacAddress),
                new OdbcParameter("ItChampionName",
(object)newRequest.ItChampionName ?? DBNull.Value),
                new OdbcParameter("RequestDate", newRequest.RequestDate),
                new OdbcParameter("Status", newRequest.Status),
                new OdbcParameter("WorkflowStage", newRequest.WorkflowStage),
                new OdbcParameter("HodComments",
(object)newRequest.HodComments ?? DBNull.Value),
                new OdbcParameter("ItChampionReviewDate",
(object)newRequest.ItChampionReviewDate ?? DBNull.Value)
            };

            // Execute insert (no result expected)
            DbHelper.ExecuteNonQuery(insertQuery, parameters);

            // 2. Get last inserted ID (in separate query)
            string idQuery = "SELECT LAST_INSERT_ID();";
            DataTable idResult = DbHelper.ExecuteQuery(idQuery, null);
            int newRequestId = Convert.ToInt32(idResult.Rows[0][0]);

            // Success response
            responseJson = jsSerializer.Serialize(new { Id = newRequestId, Message =
"Request submitted successfully." });
            context.Response.StatusCode = 200;
        }
        catch (Exception ex)
        {
            context.Response.StatusCode = 500;
            responseJson = jsSerializer.Serialize(new { Message = "Server error: " +
ex.Message });
        }
    }
```

```
            else
            {
                context.Response.StatusCode = 405;
                responseJson = jsSerializer.Serialize(new { Message = "Only POST requests are
allowed for submitting requests." });
            }

            context.Response.Write(responseJson);
        }

        public bool IsReusable
        {
            get { return false; }
        }
    }
}
```

UpdateReuqestHandler.ashx.cs

```
using System;
using System.Web;
using System.Web.Script.Serialization;
using System.Data;
using System.Data.Odbc; // Changed from MySql.Data.MySqlClient
using System.Collections.Generic; // Required for List<T>

// Make sure your DbHelper and Request classes are accessible within the ECNET.Web
namespace.

namespace ECNET.Web
{
    public class UpdateRequestHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            context.Response.ContentType = "application/json";
            string responseJson = "";
            var jsSerializer = new JavaScriptSerializer();

            if (context.Request.HttpMethod == "POST")
            {
                try
                {
                    string requestBody = new
System.IO.StreamReader(context.Request.InputStream).ReadToEnd();
                    dynamic updateData = jsSerializer.Deserialize<dynamic>(requestBody);

                    int requestId = updateData["Id"];
                    string newStatus = updateData["Status"];
```

```
            // Fix for Error 9, 11, 14: Use 'as string' or null-coalescing for potential null
values
            string hodComments = updateData["HodComments"] as string;
            string itChampionComments = updateData["ItChampionComments"] as
string;
            string workflowStage = updateData["WorkflowStage"];
            string assignedHOD = updateData["AssignedHOD"] as string;


            string query = "UPDATE Requests SET Status = ?, WorkflowStage = ?"; //
ODBC uses '?'
            List<OdbcParameter> parameters = new List<OdbcParameter>(); // Changed
from MySqlParameter
            parameters.Add(new OdbcParameter("Status", newStatus));
            parameters.Add(new OdbcParameter("WorkflowStage", workflowStage));

            // Conditionally add parameters based on who is updating
            if (!string.IsNullOrEmpty(hodComments)) // Check for actual string content
            {
                query += ", HodComments = ?";
                parameters.Add(new OdbcParameter("HodComments", hodComments));
            }
            if (!string.IsNullOrEmpty(itChampionComments)) // Check for actual string
content
            {
                query += ", ItChampionComments = ?";
                parameters.Add(new OdbcParameter("ItChampionComments",
itChampionComments));
                query += ", ItChampionReviewDate = ?"; // Set review date for IT
Champion
                parameters.Add(new OdbcParameter("ItChampionReviewDate",
DateTime.Now));
            }
            if (!string.IsNullOrEmpty(assignedHOD)) // Check for actual string content
            {
                query += ", AssignedHOD = ?"; // Assuming a column for assigned HOD in
Requests table
                parameters.Add(new OdbcParameter("AssignedHOD", assignedHOD));
            }

            query += " WHERE Id = ?"; // ODBC uses '?'
            parameters.Add(new OdbcParameter("Id", requestId));

            int rowsAffected = DbHelper.ExecuteNonQuery(query,
parameters.ToArray()); // DbHelper now expects OdbcParameter

            if (rowsAffected > 0)
            {
                responseJson = jsSerializer.Serialize(new { Message = "Request updated
successfully." });
                context.Response.StatusCode = 200;
            }
```

```
                else
                {
                    context.Response.StatusCode = 404; // Not Found or No rows affected
                    responseJson = jsSerializer.Serialize(new { Message = "Request not found
or no changes made." });
                }
            }
            catch (Exception ex)
            {
                context.Response.StatusCode = 500;
                responseJson = jsSerializer.Serialize(new { Message = "Server error: " +
ex.Message });
            }
        }
        else
        {
            context.Response.StatusCode = 405; // Method Not Allowed
            responseJson = jsSerializer.Serialize(new { Message = "Only POST requests are
allowed for updating requests." });
        }

        context.Response.Write(responseJson);
    }

    public bool IsReusable
    {
        get { return false; }
    }
  }
}
```

ViewRequestHandler.ashx.cs

```
using System;
using System.Web;
using System.Web.Script.Serialization;
using System.Data;
using System.Data.Odbc;
using System.Collections.Generic;
using System.IO;

namespace ECNET.Web
{
  public class ViewRequestHandler : IHttpHandler
  {
    public void ProcessRequest(HttpContext context)
    {
        context.Response.ContentType = "application/json";
        context.Response.AddHeader("Access-Control-Allow-Origin", "*");

        string responseJson = "";
```

```
var jsSerializer = new JavaScriptSerializer();

if (context.Request.HttpMethod == "GET")
{
    try
    {
        // Get filter parameters from query string
        string roleFilter = context.Request.QueryString["role"];
        string statusFilter = context.Request.QueryString["status"];
        string applicantNameFilter = context.Request.QueryString["applicant"];
        string departmentFilter = context.Request.QueryString["department"];
        string workflowStageFilter = context.Request.QueryString["workflowStage"];

        // Get pagination parameters
        int page = 1;
        int pageSize = 50;

        if (!string.IsNullOrEmpty(context.Request.QueryString["page"]))
            int.TryParse(context.Request.QueryString["page"], out page);

        if (!string.IsNullOrEmpty(context.Request.QueryString["pageSize"]))
            int.TryParse(context.Request.QueryString["pageSize"], out pageSize);

        // Build query with filters
        string query = "SELECT * FROM Requests WHERE 1=1";
        List<OdbcParameter> parameters = new List<OdbcParameter>();

        // Apply role-based filtering
        if (!string.IsNullOrEmpty(roleFilter))
        {
            if (roleFilter.ToLower() == "hod")
            {
                // HOD should only see requests that reached the HOD stage
                query += " AND WorkflowStage = ?";
                parameters.Add(new OdbcParameter("stage", "HOD Review"));
            }
            else if (roleFilter.ToLower() == "it" || roleFilter.ToLower() == "it-champion")
            {
                // IT Champion sees only requests that are assigned to them
                query += " AND WorkflowStage = ?";
                parameters.Add(new OdbcParameter("stage", "IT Champion Review"));
            }
        }


        // Apply other filters
        if (!string.IsNullOrEmpty(statusFilter))
        {
            query += " AND Status = ?";
            parameters.Add(new OdbcParameter("status", statusFilter));
        }
```

```
            if (!string.IsNullOrEmpty(applicantNameFilter))
            {
                query += " AND ApplicantName LIKE ?";
                parameters.Add(new OdbcParameter("applicant", "%" +
applicantNameFilter + "%"));
            }

            if (!string.IsNullOrEmpty(departmentFilter))
            {
                query += " AND Department = ?";
                parameters.Add(new OdbcParameter("department", departmentFilter));
            }

            if (!string.IsNullOrEmpty(workflowStageFilter))
            {
                query += " AND WorkflowStage = ?";
                parameters.Add(new OdbcParameter("workflowStage",
workflowStageFilter));
            }

            // Add ordering
            query += " ORDER BY RequestDate DESC";

            // Execute query
            DataTable dt = DbHelper.ExecuteQuery(query, parameters.ToArray());

            List<ViewRequestModel> requests = new List<ViewRequestModel>();
            foreach (DataRow row in dt.Rows)
            {
                requests.Add(new ViewRequestModel
                {
                    Id = Convert.ToInt32(row["Id"]),
                    RequestID = Convert.ToInt32(row["Id"]), // Add this for JavaScript
compatibility
                    ApplicantName = row["ApplicantName"].ToString(),
                    Department = row["Department"].ToString(),
                    Designation = row["Designation"].ToString(),
                    Email = row["Email"].ToString(),
                    ContactPhone = row["ContactPhone"].ToString(),
                    SystemMake = row["SystemMake"].ToString(),
                    Configuration = row["Configuration"].ToString(),
                    OsName = row["OsName"].ToString(),
                    AntivirusName = row["AntivirusName"] != DBNull.Value ?
row["AntivirusName"].ToString() : null,
                    MacAddress = row["MacAddress"].ToString(),
                    ItChampionName = row["ItChampionName"] != DBNull.Value ?
row["ItChampionName"].ToString() : null,
                    RequestDate = Convert.ToDateTime(row["RequestDate"]),
                    Status = row["Status"].ToString(),
                    HodComments = row["HodComments"] != DBNull.Value ?
row["HodComments"].ToString() : null,
                    WorkflowStage = row["WorkflowStage"].ToString(),
```

```
                    ItChampionReviewDate = row["ItChampionReviewDate"] !=
DBNull.Value ?
                        (DateTime?)Convert.ToDateTime(row["ItChampionReviewDate"]) : null
                });
            }

            var response = new
            {
                Requests = requests,
                CurrentPage = page,
                PageSize = pageSize,
                TotalRecords = requests.Count,
                FilterApplied = new
                {
                    Role = roleFilter,
                    Status = statusFilter,
                    Applicant = applicantNameFilter,
                    Department = departmentFilter,
                    WorkflowStage = workflowStageFilter
                }
            };

            responseJson = jsSerializer.Serialize(response);
            context.Response.StatusCode = 200;
        }
        catch (Exception ex)
        {
            context.Response.StatusCode = 500;
            responseJson = jsSerializer.Serialize(new
            {
                Message = "Server error: " + ex.Message,
                StackTrace = ex.StackTrace // Remove in production
            });
        }
    }
    else if (context.Request.HttpMethod == "POST")
    {
        try
        {
            string requestBody = "";
            using (StreamReader reader = new
StreamReader(context.Request.InputStream))
            {
                requestBody = reader.ReadToEnd();
            }

            if (string.IsNullOrEmpty(requestBody))
            {
                context.Response.StatusCode = 400;
                responseJson = jsSerializer.Serialize(new { Message = "Request body is
empty." });
                context.Response.Write(responseJson);
```

```
                return;
            }

            dynamic updateData = jsSerializer.Deserialize<dynamic>(requestBody);

            if (updateData == null)
            {
                context.Response.StatusCode = 400;
                responseJson = jsSerializer.Serialize(new { Message = "Invalid JSON
data." });
                context.Response.Write(responseJson);
                return;
            }

            int requestId = 0;
            string action = null;
            string hodComments = "";

            // Handle both Id and requestId for compatibility
            if (updateData.ContainsKey("Id"))
                int.TryParse(updateData["Id"].ToString(), out requestId);
            else if (updateData.ContainsKey("requestId"))
                int.TryParse(updateData["requestId"].ToString(), out requestId);

            if (updateData.ContainsKey("Action"))
                action = updateData["Action"] != null ? updateData["Action"].ToString() :
null;
            else if (updateData.ContainsKey("action"))
                action = updateData["action"] != null ? updateData["action"].ToString() :
null;

            if (updateData.ContainsKey("HodComments"))
                hodComments = updateData["HodComments"] != null ?
updateData["HodComments"].ToString() : "";
            else
                hodComments = "";

            if (requestId == 0)
            {
                context.Response.StatusCode = 400;
                responseJson = jsSerializer.Serialize(new { Message = "Invalid or missing
request ID." });
                context.Response.Write(responseJson);
                return;
            }

            if (string.IsNullOrEmpty(action))
            {
                context.Response.StatusCode = 400;
                responseJson = jsSerializer.Serialize(new { Message = "Action is required."
});
                context.Response.Write(responseJson);
```

```
                return;
            }

            string updateQuery = "";
            List<OdbcParameter> updateParameters = new List<OdbcParameter>();

            switch (action.ToLower())
            {
                case "hod_approve":
                    updateQuery = "UPDATE Requests SET Status = ?, WorkflowStage = ?,
HodComments = ? WHERE Id = ?";
                    updateParameters.Add(new OdbcParameter("status", "HOD
Approved"));
                    updateParameters.Add(new OdbcParameter("stage", "IT Champion
Review"));
                    updateParameters.Add(new OdbcParameter("comments",
hodComments));
                    updateParameters.Add(new OdbcParameter("id", requestId));
                    break;

                case "hod_reject":
                    updateQuery = "UPDATE Requests SET Status = ?, WorkflowStage = ?,
HodComments = ? WHERE Id = ?";
                    updateParameters.Add(new OdbcParameter("status", "HOD Rejected"));
                    updateParameters.Add(new OdbcParameter("stage", "Rejected"));
                    updateParameters.Add(new OdbcParameter("comments",
hodComments));
                    updateParameters.Add(new OdbcParameter("id", requestId));
                    break;

                case "it_complete":
                case "forward_to_hod":
                    updateQuery = "UPDATE Requests SET Status = ?, WorkflowStage = ?,
ItChampionReviewDate = ? WHERE Id = ?";
                    updateParameters.Add(new OdbcParameter("status", "Pending HOD
Review"));
                    updateParameters.Add(new OdbcParameter("stage", "HOD Review"));
                    updateParameters.Add(new OdbcParameter("reviewDate",
DateTime.Now));
                    updateParameters.Add(new OdbcParameter("id", requestId));
                    break;

                default:
                    context.Response.StatusCode = 400;
                    responseJson = jsSerializer.Serialize(new { Message = "Invalid action
specified: " + action });
                    context.Response.Write(responseJson);
                    return;
            }

            int affectedRows = DbHelper.ExecuteNonQuery(updateQuery,
updateParameters.ToArray());
```

```
                if (affectedRows > 0)
                {
                    responseJson = jsSerializer.Serialize(new
                    {
                        success = true,
                        Message = "Request updated successfully.",
                        AffectedRows = affectedRows,
                        RequestId = requestId,
                        Action = action
                    });
                    context.Response.StatusCode = 200;
                }
                else
                {
                    responseJson = jsSerializer.Serialize(new
                    {
                        success = false,
                        Message = "No rows were updated. Request ID may not exist or no
changes were made.",
                        RequestId = requestId
                    });
                    context.Response.StatusCode = 400;
                }
            }
            catch (Exception ex)
            {
                context.Response.StatusCode = 500;
                responseJson = jsSerializer.Serialize(new
                {
                    success = false,
                    Message = "Server error: " + ex.Message,
                    StackTrace = ex.StackTrace // Remove in production
                });
            }
        }
        else
        {
            context.Response.StatusCode = 405;
            responseJson = jsSerializer.Serialize(new { Message = "Only GET and POST
requests are allowed." });
        }

        context.Response.Write(responseJson);
    }

    public bool IsReusable
    {
        get { return false; }
    }
}
```

```
    public class ViewRequestModel
    {
        public int Id { get; set; }
        public int RequestID { get; set; } // Add this for JavaScript compatibility
        public string ApplicantName { get; set; }
        public string Department { get; set; }
        public string Designation { get; set; }
        public string Email { get; set; }
        public string ContactPhone { get; set; }
        public string SystemMake { get; set; }
        public string Configuration { get; set; }
        public string OsName { get; set; }
        public string AntivirusName { get; set; }
        public string MacAddress { get; set; }
        public string ItChampionName { get; set; }
        public DateTime RequestDate { get; set; }
        public string Status { get; set; }
        public string HodComments { get; set; }
        public string WorkflowStage { get; set; }
        public DateTime? ItChampionReviewDate { get; set; }
    }
}
```