How to create a benchmark concept for a final thesis in case of deriving IT-security (risk) index values including test-challenges based on Copilot

László Pitlik (MY-X team)

Content

[Introduction 1](#_Toc190592178)

[Layers 1](#_Toc190592179)

[1. Layer: Index-values (benchmarking) – with critical aspects 1](#_Toc190592180)

[2. Layer: automations – with critical aspects 3](#_Toc190592181)

[3. Layer: Testing – with critical aspects 9](#_Toc190592182)

[HTML 9](#_Toc190592183)

[Javascript 11](#_Toc190592184)

[Conclusions 12](#_Toc190592185)

# Introduction

The following case study presents a conversation process with a COPILOT version in order to demonstrate how can we create a benchmark (first layer) for our own final thesis. The own solution should always be become better than this benchmark.

On the other hand (second layer), we are searching for automation-aspects behind the benchmark concept in order to collect ideas what part can be automated in the own concept.

Parallel (third layer), the automated part should always be tested: the COPILOT-conversation should deliver details about the testing characteristics.

# Layers

## Layer: Index-values (benchmarking) – with critical aspects

Prompt:

Basic information: in the world, there is a lot of home workers. Quasi each of them has (even dynamically) different circumstances concerning his/her IT-security risks. Please, create a concept for a software being capable of estimating an IT security (risk) index value for arbitrary home workers! This concept should define at least 10 attributes for describing IT-security risk-layers! The concept should also present the calculation scheme based on the raw attribute values which leading finally to the expected risk index value in case of concrete raw values.

Copilot:

Sure, I'd be happy to help with that! Here's a concept for a software that estimates an IT security risk index for home workers:

**Concept for IT Security Risk Index Software**

**Attributes for Describing IT-Security Risk Layers (Critical aspects: e.g. there are never given measurement units! There are never given direction rules: e.g. the younger is antivirus version the lower is IT-security risk for a home worker…)**

1. **Device Security**: Measures the security features of the devices used (e.g., antivirus, firewall, encryption).
2. **Network Security**: Assesses the security of the home network (e.g., router security, use of VPN).
3. **Software Updates**: Frequency and timeliness of software updates and patches.
4. **User Awareness**: Level of user knowledge and awareness about IT security practices.
5. **Access Control**: Measures the strength of access controls (e.g., passwords, multi-factor authentication).
6. **Data Backup**: Frequency and reliability of data backups.
7. **Physical Security**: Security of the physical environment (e.g., locked rooms, secure storage of devices).
8. **Incident Response**: Preparedness and response plan for IT security incidents.
9. **Remote Access Security**: Security measures for remote access (e.g., secure remote desktop protocols).
10. **Third-Party Risk**: Security of third-party services and applications used.

**Calculation Scheme**

The software will assign a score to each attribute based on raw values provided by the user. Each attribute will have a weight reflecting its importance to overall IT security. The final IT security risk index will be a weighted average of these scores.

**Example Calculation (Critical aspects: Scores are never measured data, it means: the definitions above did not explain how a raw score should be observed/measured/derived? Weights are totally subjective inputs for a calculation scheme… Directions for the attributes are not given…)**

1. **Device Security**: Score = 8 (out of 10), Weight = 0.15
2. **Network Security**: Score = 7, Weight = 0.15
3. **Software Updates**: Score = 6, Weight = 0.10
4. **User Awareness**: Score = 5, Weight = 0.10
5. **Access Control**: Score = 9, Weight = 0.10
6. **Data Backup**: Score = 7, Weight = 0.10
7. **Physical Security**: Score = 8, Weight = 0.10
8. **Incident Response**: Score = 6, Weight = 0.10
9. **Remote Access Security**: Score = 7, Weight = 0.05
10. **Third-Party Risk**: Score = 5, Weight = 0.05

The final IT security risk index (R) is calculated as follows:

$ R = \sum*{i=1}^{10} (\text{Score}*i \times \text{Weight}\_i) $

Using the example scores and weights:

$ R = (8 \times 0.15) + (7 \times 0.15) + (6 \times 0.10) + (5 \times 0.10) + (9 \times 0.10) + (7 \times 0.10) + (8 \times 0.10) + (6 \times 0.10) + (7 \times 0.05) + (5 \times 0.05) $

$ R = 1.2 + 1.05 + 0.6 + 0.5 + 0.9 + 0.7 + 0.8 + 0.6 + 0.35 + 0.25 = 6.95 $

The IT security risk index for this example is **6.95** out of 10.

Would you like to dive deeper into any specific attribute or calculation detail?

Conclusion: the concept is a totally “empty bubble” (without any real/concrete meaning): a potential programmer may never be able to create any source code based e.g. on real log-data about the circumstances of a concrete home worker…

## Layer: automations – with critical aspects

Prompt:

Please, create html-code for collecting the attribute-values based on predefined score-lists and please create javascript-code for the button which is responsible for the calculation as such!

Copilot:

Sure! Here's an example of HTML and JavaScript code to collect attribute values based on predefined score lists and calculate the IT security risk index:

**HTML Code**

<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <title>IT Security Risk Index Calculator</title>

</head>

<body>

 <h1>IT Security Risk Index Calculator</h1>

 <form id="riskForm">

 <label for="deviceSecurity">Device Security:</label>

 <select id="deviceSecurity">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="networkSecurity">Network Security:</label>

 <select id="networkSecurity">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="softwareUpdates">Software Updates:</label>

 <select id="softwareUpdates">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="userAwareness">User Awareness:</label>

 <select id="userAwareness">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="accessControl">Access Control:</label>

 <select id="accessControl">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="dataBackup">Data Backup:</label>

 <select id="dataBackup">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="physicalSecurity">Physical Security:</label>

 <select id="physicalSecurity">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="incidentResponse">Incident Response:</label>

 <select id="incidentResponse">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="remoteAccessSecurity">Remote Access Security:</label>

 <select id="remoteAccessSecurity">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <label for="thirdPartyRisk">Third-Party Risk:</label>

 <select id="thirdPartyRisk">

 <option value="8">8</option>

 <option value="7">7</option>

 <option value="6">6</option>

 <option value="5">5</option>

 <option value="4">4</option>

 <option value="3">3</option>

 <option value="2">2</option>

 <option value="1">1</option>

 <option value="0">0</option>

 </select><br>

 <button type="button" onclick="calculateRisk()">Calculate Risk Index</button>

 </form>

 <p id="result"></p>

 <script>

 function calculateRisk() {

 const weights = {

 deviceSecurity: 0.15,

 networkSecurity: 0.15,

 softwareUpdates: 0.10,

 userAwareness: 0.10,

 accessControl: 0.10,

 dataBackup: 0.10,

 physicalSecurity: 0.10,

 incidentResponse: 0.10,

 remoteAccessSecurity: 0.05,

 thirdPartyRisk: 0.05

 };

 const scores = {

 deviceSecurity: parseInt(document.getElementById('deviceSecurity').value),

 networkSecurity: parseInt(document.getElementById('networkSecurity').value),

 softwareUpdates: parseInt(document.getElementById('softwareUpdates').value),

 userAwareness: parseInt(document.getElementById('userAwareness').value),

 accessControl: parseInt(document.getElementById('accessControl').value),

 dataBackup: parseInt(document.getElementById('dataBackup').value),

 physicalSecurity: parseInt(document.getElementById('physicalSecurity').value),

 incidentResponse: parseInt(document.getElementById('incidentResponse').value),

 remoteAccessSecurity: parseInt(document.getElementById('remoteAccessSecurity').value),

 thirdPartyRisk: parseInt(document.getElementById('thirdPartyRisk').value)

 };

 let riskIndex = 0;

 for (const key in scores) {

 riskIndex += scores[key] \* weights[key];

 }

 document.getElementById('result').innerText = `IT Security Risk Index: ${riskIndex.toFixed(2)} out of 10`;

 }

 </script>

</body>

</html>

This code provides a form with dropdowns for each attribute, allowing users to select a score from 0 to 8. When the "Calculate Risk Index" button is clicked, the JavaScript function calculateRisk() calculates the weighted average of the scores and displays the IT security risk index.

Feel free to customize the score ranges, weights, and other details as needed! If you have any questions or need further adjustments, just let me know.

Conclusion: the general logic seems to be rational, but the code should be capable of running!

The weights are predefined – because the prompt said no information about its handling…

## Layer: Testing – with critical aspects

### HTML

Prompt:

Please, create a testing concept for the previous html solution: which critical aspects should be tested and how?

Copilot:

Certainly! Here's a testing concept for the HTML solution, focusing on critical aspects to ensure the functionality and reliability of the IT Security Risk Index Calculator:

**Testing Concept**

**1. Form Validation**

* **Objective**: Ensure that all form elements accept valid inputs and handle invalid inputs gracefully.
* **Tests**:
	+ Verify that each dropdown allows selection of values from 0 to 8.
	+ Check that the form does not submit if any dropdown is left unselected (if required).
	+ Ensure that the form resets correctly after submission.

**2. Calculation Accuracy (Critical aspects: It is not directly a HTML-code-test, it is already a test for the javascript-part)**

* **Objective**: Confirm that the risk index calculation is accurate based on the selected scores and predefined weights.
* **Tests**:
	+ Manually calculate the risk index for a set of known inputs and compare it with the output from the calculator.
	+ Test edge cases, such as all scores being 0 or 8, to ensure the calculation handles these correctly.

**3. User Interface (UI)**

* **Objective**: Ensure the UI is user-friendly and displays information correctly.
* **Tests**:
	+ Verify that all labels and dropdowns are correctly aligned and readable.
	+ Check that the calculated risk index is displayed in the correct format and location.
	+ Ensure that the "Calculate Risk Index" button is functional and responsive.

**4. Error Handling**

* **Objective**: Ensure the application handles errors gracefully without crashing.
* **Tests**:
	+ Simulate scenarios where JavaScript might fail (e.g., missing elements) and ensure appropriate error messages are displayed.
	+ Verify that the application does not break if unexpected values are selected or entered.

**5. Cross-Browser Compatibility**

* **Objective**: Ensure the application works consistently across different web browsers.
* **Tests**:
	+ Test the application on major browsers (e.g., Chrome, Firefox, Safari, Edge) to ensure consistent behavior.
	+ Verify that the layout and functionality remain intact across different browsers.

**6. Performance**

* **Objective**: Ensure the application performs efficiently without significant delays.
* **Tests**:
	+ Measure the time taken to calculate and display the risk index after clicking the button.
	+ Ensure that the application remains responsive even with multiple form submissions.

**7. Security**

* **Objective**: Ensure the application is secure and does not expose vulnerabilities.
* **Tests**:
	+ Verify that the form does not allow script injections or other malicious inputs.
	+ Ensure that sensitive data (if any) is handled securely.

**Testing Steps**

1. **Unit Testing**: Test individual components (e.g., dropdowns, calculation function) in isolation to ensure they work correctly.
2. **Integration Testing**: Test the interaction between components to ensure they work together as expected.
3. **System Testing**: Test the entire application as a whole to ensure it meets the requirements and performs correctly.
4. **User Acceptance Testing (UAT)**: Have end-users test the application to ensure it meets their needs and is user-friendly.

By following this testing concept, you can ensure that the IT Security Risk Index Calculator is reliable, accurate, and user-friendly. If you need further details or specific test cases, feel free to ask!

### Javascript

Prompt:

The previous prompt has already triggered the Copilot to deliver a test concept for the javascript part too.

Copilot:

n/a

# Conclusions

A good final thesis should at least be capable of performing each above-mentioned aspects where the critical aspects are already finetuned!