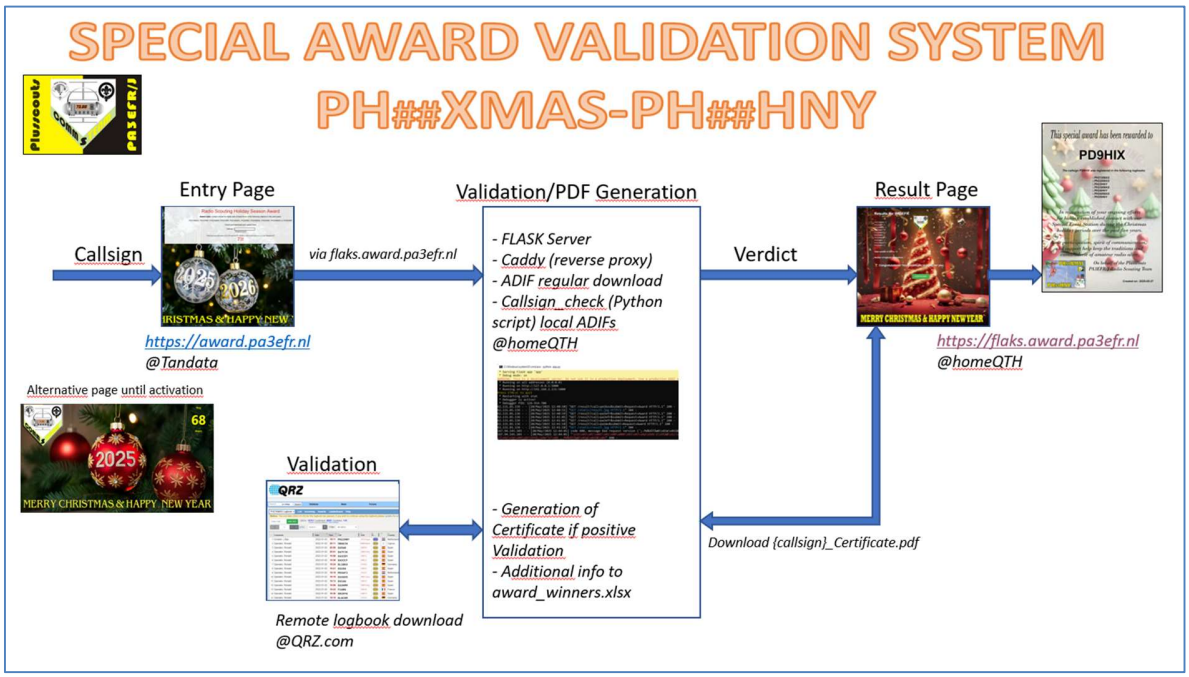
Special Award Validation System (SAVS)

for PH##XMAS and PH##HNY



Rijswijk, September 2025

PA3EFR

INTRODUCTION

Since we wanted to do something extra during the anniversary year of PH##XMAS/PH##HNY—beyond just making QSOs—we thought it would be fun to grant an award to loyal radio operators who have worked with us several times in the past. We also wanted to make this process fully automated through a website.

Using Python programming skills, AI tools, and a great deal of patience, the following system was developed: the Special Award Validation System (SAVS).

This document describes how it works, provides the scripts, and explains the methods that make up SAVS.

Chapters 1 and 2 describe each of the basic components in detail, as well as the minimum required servers, functionalities, and applications. Chapter 3 discusses the challenges encountered and the solutions implemented.

The appendices include the various batch files and Python scripts. These are not my private property and may be freely reused for educational or recreational purposes.

CHAPTER 1 BASIC COMPONENTS

SAVS consists of the following components and their functions:

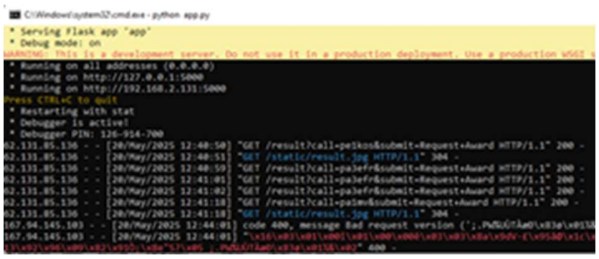
1. Entry Page

This page is the first entry point for an operator to check whether he or she qualifies for an award. The URL for this page is https://award.pa3efr.nl, hosted on a commercial server as a subdomain of pa3efr.nl (in my case, Tandata).

On this page, an input field allows the operator to submit their callsign to check whether they meet the required number of QSOs over the past five years (including 2025–2026). The request is sent to a server running on my ThinClient, which also hosts WIRES-X.

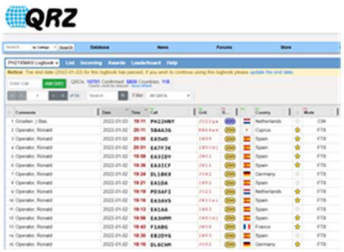
Until the activation date of SAVS is reached, the site displays a countdown page to December 6—the date when PH##XMAS will first go on air.

2. The Flask Server



This server receives the callsign from the Entry Page and forwards the request to a Python script that verifies it against the PH##XMAS and PH##HNY logs. ADIF files from previous years are preloaded, and the ADIF of the current year is automatically downloaded from QRZ.com every 15 minutes.

Because the Entry Page runs in a secure environment (HTTPS), the Flask server is also secured through a Reverse Proxy configured using a Caddyfile. This provides a valid Let's Encrypt certificate and resolves mixed environment issues.



If the request meets the criteria, the Python script returns a positive validation on a Result Page, displaying all confirmed logs (as an overview). Simultaneously,

a certificate is generated, and the award_winners Excel file is updated to register the issued award.

3. Result Page

On the Result Page, the operator can see in which years QSOs were made that qualify for the award. This page runs on the Flask server at my home.

A button at the bottom allows the operator to download the certificate as a PDF, clearly showing the confirmed results.

4. Award/Certificate

The certificate shows the years in which the callsign appeared in the checked logbooks. This contributes to operator engagement and proudly recognizes their effort to earn the award during the anniversary year.

Upon request, the certificate can also be retrieved after the event, as a copy (PDF) is stored on the ThinClient, along with the Excel registration.

CHAPTER 2 FUNCTIONALITIES

This chapter provides detailed descriptions of the different applications and components. The appendices contain reusable batch files and scripts.

The following sections are described in sequence:

1. Entry Page
2. ThinClient (TC) with:
    a. Flask Server
    b. Caddyfile
    c. Logbook Download
    d. Callsign Check Script
    e. Certificate Generator
    f. Award Winner Registration
3. Result Page

Bookmarks are provided where possible for easy navigation between chapters, sections, and appendices.

Section 1: Entry Page

Before December 6, the Entry Page looks different than after. This is because the validation server is inactive until that date. Starting December 6, PH##XMAS becomes active, and operators can check whether they qualify for an award.

This operational award page (https://award.pa3efr.nl) is the first interface in the award process. The operator enters their callsign to check eligibility.

The text indicates that at least three of the listed stations must have been logged in past years.

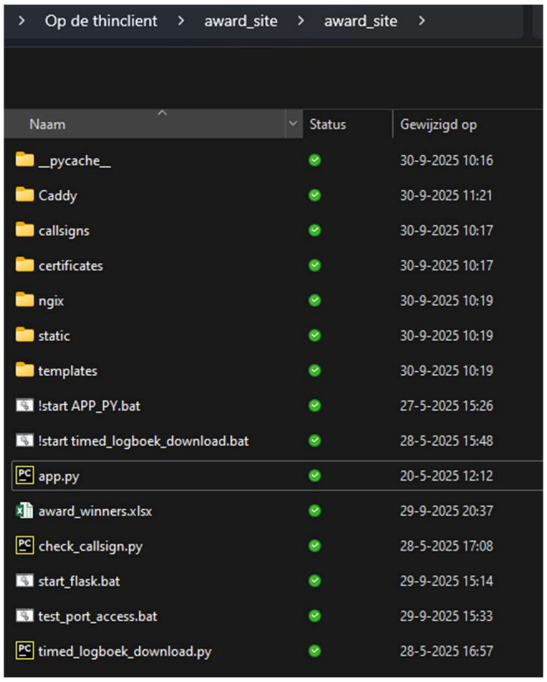
A small note warns that the validation may take a few seconds to complete, but this is generally not a problem.

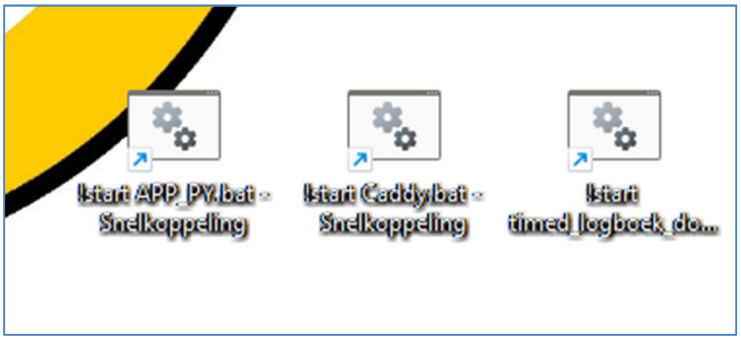
The structure and HTML content of this page (hosted by Tandata) can be found in Appendix A.

Section 2: ThinClient (TC)

The ThinClient used for this award runs permanently in my radio shack for WIRES-X. For energy efficiency, this low-power device was chosen to handle award validations during the holiday period.

It runs Windows 11 Pro with a local hard drive and hosts WIRES-X, the Flask server, and supporting software (Caddyfile, etc.) for processing award requests.





Section 2a: Flask Server

A Flask server processes incoming user requests and returns structured responses. It acts as the bridge between the front end (Entry Page) and the backend validation script (check_callsign.py) running on the TC.

The Flask server must be started first by executing `app.py` via the batch file `!start APP_PY.bat`.

Scripts for both are included in Appendices B and C.

The script checks whether a callsign qualifies for an award (via check_callsign.py; see Section 2d) and returns the result to `result.html` (see Section 2e). This HTML template is located in the “templates” directory and uses images from the “static” folder.

Finally, the script generates a certificate through the `download_certificate` routine and saves it as a PDF under the operator’s callsign (see Section 2f).

Section 2b: Caddyfile

A Caddyfile acts as a reverse proxy, meaning it receives incoming requests from clients (the Entry Page) and forwards them to the backend Flask server.

In a mixed security environment (some services HTTPS, others HTTP), Caddy ensures that all external traffic runs securely over HTTPS while allowing internal flexibility between both secure and non-secure backends. It therefore provides a safe and consistent interface to users.

This allows the secure Entry Page to communicate with the non-secure Result Page.

Caddy automatically retrieves a Let’s Encrypt certificate at set intervals to present to the Entry Page during requests.

Caddy configuration files are stored in the Caddy directory and detailed in Appendix D. The file is started second via the `!start Caddy.bat` routine.

Section 2c: Logbook Download

Before any callsign can be checked, several logbooks are required:

- Logbooks from previous years (downloaded manually as ADIF files).

- Current-year logbooks, automatically fetched from QRZ.com at defined intervals using API keys.

The script `timed_logboek_download.py` performs this task and is launched as the third process using `!start timed_logboek_download.bat`.

Each run triggers a countdown timer until the next update (default interval: 15 minutes).

See Appendix E for the scripts.

Section 2d: Callsign Check Script

The script `check_callsign.py` verifies whether a callsign appears in the ADIF logs of past and current years. It is a relatively simple check that parses the available ADIF files (including those downloaded from QRZ.com).

See Appendix F for details.

Section 2e: Certificate Generator

Within the same `check_callsign.py` script, a second function generates a certificate for callsigns that meet the award requirements.

Generated certificates are stored in the “certificates” folder as `<callsign>_certificate.pdf`.

Section 2f: Award Winner Registration

When an operator qualifies for an award, the information is recorded in `award_winners.xlsx`. This is part of `check_callsign.py`. It is recommended to clean up this file before the event to remove test entries.

Callsign	Logboeken	Datum van vandaag	Aantal logboeken	Aantal QSO's

Section 3: Result Page

After submitting a callsign on the Entry Page, the operator receives a new window displaying their results—the number of valid logbook entries and whether they can download the certificate.

This page includes a routine that retrieves the appropriate PDF from the “certificates” folder for download.

The HTML file `result.html` is located in the “templates” directory, using images from “static.” Its content is shown in Appendix G.

If fewer than three qualifying contacts exist, this message is displayed, and no certificate is available for download.

CHAPTER 3 CHALLENGES AND SOLUTIONS

During the development of SAVS, I encountered the following challenges:

1. Joomla vs HTML

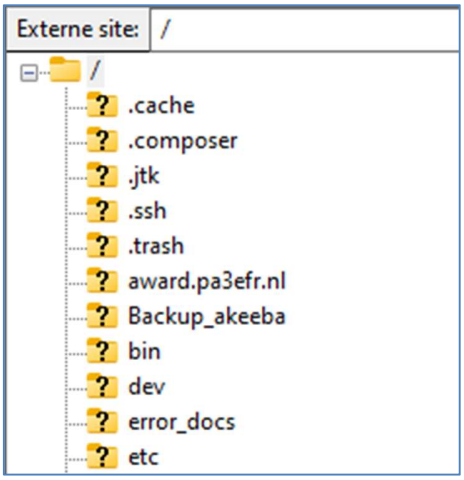
2 HTTPS vs HTTP

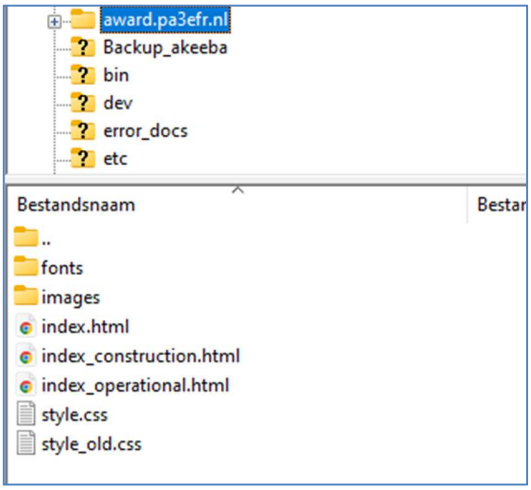
3 Data processing on a static website and static PDF

Below I explain the solutions found for each. The greatest assistance came from AI tools—specifically ChatGPT and agent.minimax.io.

1. Joomla vs HTML

My main website (pa3efr.nl) is built in Joomla, while the award system required plain HTML pages. Using the Plesk application, I created a subdomain at Tandata named award.pa3efr.nl. Through the FileZilla FTP application, I uploaded the HTML





files to this subdomain, making them externally accessible.

2. HTTPS vs HTTP

This challenge took the most time to resolve. With help from AI—where I described the issue, tested suggested solutions, and analyzed returned error codes—it was finally solved using Caddy as a reverse proxy.

3. Data processing on a static website and PDF

Once the `check_callsign` script determines whether an operator qualifies, the software must return their callsign and logbook results both in a browser view and as a PDF.

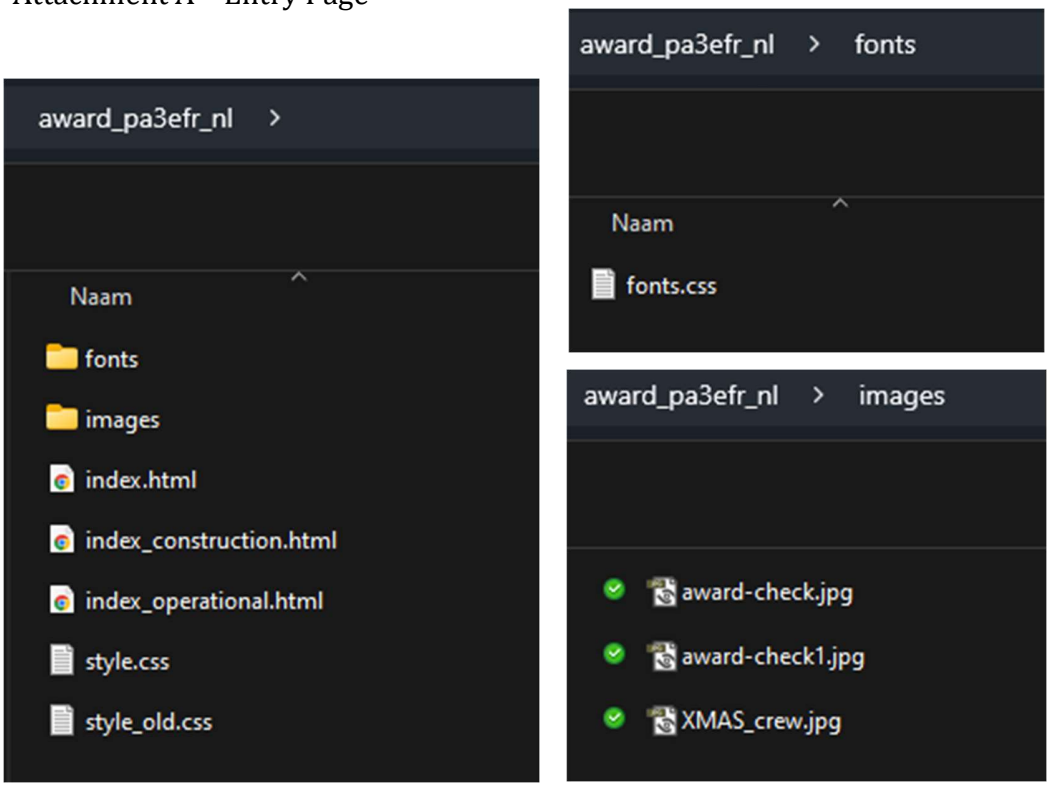
These scripts required trial and error to properly display the data. With AI assistance, I was able to complete this successfully—even without deep programming knowledge.

Overall, this was a fun project. The existing SAVS can easily be adapted for future events and may inspire others to take on similar challenges.

Erwin, PA3EFR

30 September 2025

Attachment A – Entry Page



- Index.html (index_construction.html)

```
<!doctype html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>XMAS countdown</title>
  <style>
    /* Full-page fixed background from images/XMAS_crew.jpg */
    html,body{height:100%;margin:0}
    body{
      background-image: url('https://award.pa3efr.nl/images/XMAS_crew.jpg');
      background-size: cover;
      background-position: center;
      background-repeat: no-repeat;
      background-attachment: fixed;
      -webkit-font-smoothing:antialiased;
      -moz-osx-font-smoothing:grayscale;
      user-select:none;
    }

    /* Position countdown top right but moved left between ornaments */
    #container{
```

```
      position:fixed;
      top:20px;
      right:100px; /* push it left from the edge */
      text-align:center;
      color: yellow;
      font-family: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica
Neue", Arial;
      text-shadow: 0 2px 12px rgba(0,0,0,0.7);
    }

    #label-top, #label-bottom {
      font-size: 1.5rem;
      font-weight:600;
      display:block;
    }

    #countdown{
      font-weight:700;
      font-size: calc(48px + 4vw);
      line-height:1;
      margin:0.25em 0;
    }

    /* Hide other accidental elements */
    a, img, p, h1, h2, h3, h4, h5, h6 { display: none !important; }
  </style>
</head>
<body>
  <div id="container" aria-label="Dagen tot 6 december 2025">
    <span id="label-top">Only</span>
    <div id="countdown"></div>
    <span id="label-bottom">days 2 go</span>
  </div>

  <script>
    (function(){
      // Target date: 6 December 2025 at 00:00:00 local time
      const target = new Date(2025, 11, 6, 0, 0, 0, 0); // month is 0-based: 11
= December
      const el = document.getElementById('countdown');

      function update(){
        const now = new Date();
        const msPerDay = 24*60*60*1000;
        const diff = target - now;
        let days = Math.ceil(diff / msPerDay);
        if (days < 0) days = 0;
        el.textContent = String(days);
      }

      // initial render and then update every 30 seconds
      update();
      setInterval(update, 30_000);
    })();
  </script>
</body>
</html>
```

- Index_operational.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta charset="utf-8">
    <title>Download Award</title>
    <link rel="icon" href="../favicon.ico">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="../fonts/fonts.css">

    <!-- Toegevoegde achtergrondstijl (enige wijziging) -->
    <style>
        html, body {
            height: 100%;
            margin: 0;
            padding: 0;
        }

        body {
            background-image: url('images/award-check.jpg'); /* Zorg dat dit
pad klopt */
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            min-height: 100vh;
        }
    </style>
</head>
<body>
    <div id="content">
        <p>
            <span class="h">Radio Scouting Holiday Season Award</span><br>
        </p>
        <p><b>Award rules:</b> contact should be made with at least three of
the following stations in the past years:</p>
        <p>
            PH21XMAS, PH22HNY, PH22XMAS, PH23HNY, PH23XMAS, PH24HNY, PH24XMAS,
PH25HNY, PH25XMAS or PH26HNY.
            <br><br>
            Check and download your award here:
        </p>
        <p>
            <form action="https://Flaks.award.pa3efr.nl:8443/result"
method="GET" target="_blank">
                <label for="call">Callsign:</label>
                <input type="text" id="call" name="call" pattern="[A-Za-z0-
9/]+" title="Only uppercase letters, numbers, and / are allowed." required>
                <br>
                <input type="submit" name="submit" value="Request Award">
            </form>
        </p>
        <p>
            <small><i>Checking logbooks takes up to 30 seconds for validation
and is executed on a private RaspberryPi.</i></small>
            <br>
            <span class="big">73!</span>
```

```
        </p>
    </div>
</body>
</html>
```

- Styles.css

```
body {
  background: url(images/award-check.jpg) repeat;
  text-align: center;
  margin:5%;
  background-size: cover;
}

.message {
padding:3px;
font-weight:700;
}

.big {
font-size:55px; margin:0
}

#content{
  border: 1px solid #fcfcfc;
  /*background-color: white;*/
  /*opacity:0.9;*/
  /* use background instead of opacity, so images will show without opacity */
  background: rgba(255, 255, 255, 0.90);
  border-radius: 5px;
  justify-content: center;
  /*https://getcssscan.com/css-box-shadow-examples*/
  box-shadow: rgba(0, 0, 0, 0.12) 0px 1px 3px, rgba(0, 0, 0, 0.24) 0px 1px 2px;
  overflow:hidden; /*iphone*/
  margin-left:28vw; max-width:66vw;
  padding-left:15px;
}



.thumbnail {
max-width:350px;
height:auto;
}

.thumbnail-test {
border: 1px solid black;
}

/*SSTV iframe*/
#sstv, #qso, #qrz, #awards{
max-width:1280px;
height:520px;
width:100%;
overflow:hidden !important;
}

#qrz{
height:552px;
```

```
}
.internetnl{
max-height:25px;
width:auto;
}

iframe { border: none; }

a {
text-decoration:none;
}

big {
font-size:55px;
}

.xiegu {
float:left;
padding:25px;
}

.left {
  text-align: left;
}

/*SSTV iframe*/
@media only screen and (max-width: 800px) {
   #content{
   margin-left:0; max-width:90vw;
   }
   .sstv, #sstv {
      max-width:80vw;
   }
   .xiegu {
      display: none;
   }
   .left {
      text-align: center;
   }
   body {
      background-size: 100%; /*remove after easter*/
   }
}

.pc1kshack {
width:90%;
max-width:700px;
height:auto;
border: 1px solid #cccccc;
border-radius: 5px;
}

.qrzaward {
width:130px;
height:auto;
border-radius: 5px;
background-color: white;
margin:3px;
border: 1px solid #cccccc;
}

.korenaer {
max-width:90%;
```

```
max-height: 500px;
width: auto;
}
```

- Fonts.css

```
@font-face{font-family:'alata';src:url('alata-regular-ham.woff2')
format('woff2');}

@font-face{font-family:'dejavusans';src:url('NotoSans-Regular.woff2')
format('woff2');}

@font-face{font-family:'NotoSans';src:url('NotoSans-Regular.woff2')
format('woff2');}

@font-face{font-family:'NotoSansBold';src:url('NotoSans-Bold.woff2')
format('woff2');}

@font-face{font-family:'NotoSansMono';src:url('NotoSansMono-
VariableFont_wdth,wght.woff2') format('woff2');}

/* Flag font `TwemojiCountryFlags.woff2` generated by using `npm install
country-flag-emoji-polyfill` copied woff2 file into this folder. */

@font-face{font-family:'twemoji';src:url('TwemojiCountryFlags.woff2')
format('woff2');}



body {
font-family:NotoSans,sans-serif;
}

.bold, th{
font-family:NotoSansBold,sans-serif;;
}

.h, .big {
  font-family:alata,sans-serif;
  font-size:39px;
  color:#cb2821;
}

.red {
  color:#cb2821;
}

.flag {
font-family:twemoji;
}

/* Media query to exclude Apple devices (Safari on iOS and macOS) */
@supports (-webkit-touch-callout: none) {
    .flag {
        font-family: sans-serif; /* Fallback font for Apple devices, IOS
rendering bug */
    }
}
```

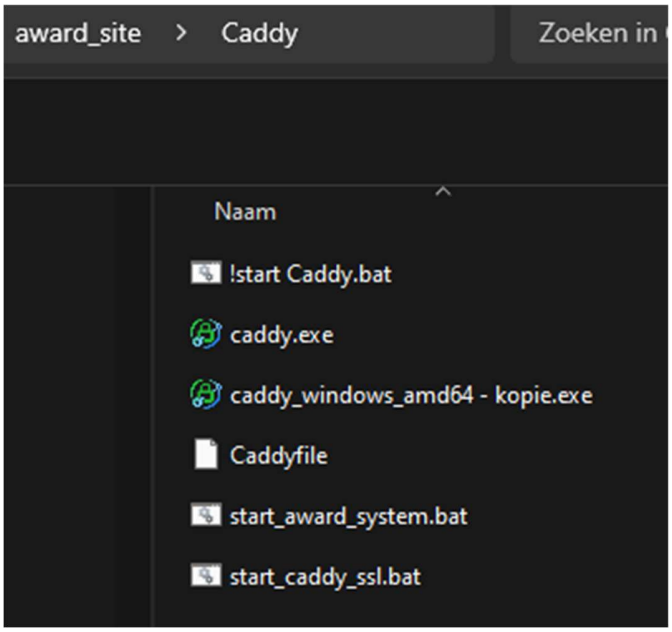
Attachment B, APP_PY.bat

```
from flask import Flask, request, render_template, send_from_directory
from check_callsign import check_callsign
import os

app = Flask(__name__)

@app.route('/')
def index():
    return '☑ Flask backend draait. Gebruik <code>/result?call=PA3EFR</code>
om een award op te vragen.'

@app.route('/result', methods=['GET'])
def result():
    call = request.args.get('call', '').strip().upper()
    if not call:
        return "❌ Geen geldige callsign opgegeven."

    data = check_callsign(call)
    return render_template('result.html', data=data)

@app.route('/certificate/<callsign>')
def download_certificate(callsign):
    cert_dir = 'certificates'
    filename = f"{callsign.upper()}_certificate.pdf"
    filepath = os.path.join(cert_dir, filename)
    if os.path.exists(filepath):
        return send_from_directory(cert_dir, filename, as_attachment=True)
    return f"❌ Certificate not found for {callsign}", 404

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Attachment C: !start APP_PY.bat

```
@echo off
START CMD /T:e0 /K "CD D:\award_site & python app.py"
```

Attachment D: Caddy Files



De files die hieronder niet beschreven zijn werden aangemaakt in de ontwikkel fase van het project. Deze worden ook niet verder uitgelegd.

- !start Caddy.bat

```
@echo off
title Award Systeem - Caddy op Poort 8443
echo ======================================
echo     Award Systeem - HTTPS Poort 8443
echo ======================================
echo.
echo Controleren of Flask server draait...
netstat -an | findstr ":5000" > nul
if errorlevel 1 (
    echo [FOUT] Flask server draait niet op poort 5000!
    echo.
    echo Start eerst Flask in een ander venster:
    echo   python app.py
    echo.
    pause
    exit /b 1
)
echo [OK] Flask server gevonden op poort 5000
echo.
echo Starten van Caddy op poort 8443...
echo.
echo De URL wordt: https://flask.award.pa3efr.nl:8443
echo.
caddy run --config Caddyfile
echo.
```

```
echo Caddy gestopt.
pause
```

- Caddyfile

```
flask.award.pa3efr.nl:8443 {
    reverse_proxy localhost:5000
}
```

Attachment E: Retrieve logbook current year

- !start timed_logbook_download.bat

```
@echo off
START CMD /T:e0 /K "CD D:\award_site & python timed_logbook_download.py"
```

- timed_logbook_download.py

```
import os
import time
import requests
import html
import urllib.parse
from datetime import datetime, timedelta

def update_remote_logbooks():
    remote_logbooks = {
        "PH25XMAS": "4DB9-…..-579F",
        "PH26HNY":  "C822-…..-0906"
    }

    adif_directory = r"D:\award_site\callsigns"

    for logbook_name, api_key in remote_logbooks.items():
        params = {
            "KEY": api_key,
            "ACTION": "FETCH",
            "ADIF": 1
        }

        try:
            response = requests.get("https://logbook.qrz.com/api",
params=params)
            if response.status_code != 200:
                print(f"❌ Fout bij ophalen {logbook_name}:
{response.status_code}")
                continue

            decoded = html.unescape(response.text)
            parsed = urllib.parse.parse_qs(decoded)
            adif_data = parsed.get("ADIF", [""])[0]

            filename = os.path.join(adif_directory, f"{logbook_name}.adi")
            with open(filename, "w", encoding="utf-8") as f:
                f.write(adif_data)
            print(f"☑ {logbook_name} opgeslagen als lokaal .adi bestand")

        except Exception as e:
            print(f"❌ Fout bij {logbook_name}: {e}")

if __name__ == "__main__":
    try:
        interval_minutes = input("Hoe vaak moet de update uitgevoerd worden (in
minuten)? [standaard: 15]: ")
```

```
        interval_minutes = int(interval_minutes) if interval_minutes.strip()
else 15
    except ValueError:
        print("⚠ Ongeldige invoer, standaard wordt 15 minuten gebruikt.")
        interval_minutes = 15

    interval = timedelta(minutes=interval_minutes)
    next_run = datetime.now()

    print(f"🔁 Script gestart. Updates elke {interval_minutes} minuten. Druk op
CTRL+C om te stoppen.")

    try:
        while True:
            if datetime.now() >= next_run:
                print(f"\n🕒 {datetime.now().strftime('%H:%M:%S')} - Update
gestart")
                update_remote_logbooks()
                next_run = datetime.now() + interval

            # Toon afteller
            remaining = (next_run - datetime.now()).total_seconds()
            minutes = int(remaining) // 60
            seconds = int(remaining) % 60
            print(f"⏳ Nog {minutes:02d}:{seconds:02d} tot volgende update",
end="\r")
            time.sleep(1)

    except KeyboardInterrupt:
        print("\n🔘 Script gestopt door gebruiker.")
```

Attachment F: Callsign check

- check_callsign.py

```
import os
import re
import html
import requests
import urllib.parse
from openpyxl import Workbook, load_workbook
from datetime import datetime
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
from reportlab.lib.utils import ImageReader
from PIL import Image

# === CONFIG ===
adif_directory = r"D:\award_site\callsigns"

allowed_local_logbooks = {
    "PH21XMAS", "PH22HNY", "PH22XMAS", "PH23HNY", "PH23XMAS",
    "PH24HNY", "PH24XMAS", "PH25HNY", "PH25XMAS", "PH26HNY"
}
call_regex = re.compile(r"<call:\d+>([A-Z0-9/]+)", re.IGNORECASE)

def check_callsign(input_call):
    input_call = input_call.strip().upper()
    results = {}

    # Lokale ADIF-bestanden
    adif_files = [f for f in os.listdir(adif_directory) if
f.lower().endswith(".adi")]
    for adif_file in adif_files:
        logbook_name = adif_file.split('.')[0].upper()
        if logbook_name not in allowed_local_logbooks:
            continue

        file_path = os.path.join(adif_directory, adif_file)

        try:
            with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
                contents = f.read()
                calls_found = call_regex.findall(contents)
                match_count = [c.upper() for c in
calls_found].count(input_call)
                results[logbook_name] = match_count

        except Exception as e:
            results[logbook_name] = f"FOUT: {e}"



    total_qsos = sum(count for count in results.values() if isinstance(count,
int))
    logbooks_with_qsos = sum(1 for count in results.values() if
isinstance(count, int) and count > 0)
    qualified = logbooks_with_qsos >= 3

    # Excel-output
    if qualified:
```

```
        excel_path = "award_winners.xlsx"
        if os.path.exists(excel_path):
            wb = load_workbook(excel_path)
            ws = wb.active
        else:
            wb = Workbook()
            ws = wb.active
            ws.append(["Callsign", "Logboeken", "Datum", "Aantal logboeken",
"Aantal QSO's"])

        logboeken_met_match = [log for log, count in results.items() if
isinstance(count, int) and count > 0]
        logboeken_str = ", ".join(logboeken_met_match)
        vandaag = datetime.today().strftime('%Y-%m-%d')

        ws.append([input_call, logboeken_str, vandaag, logbooks_with_qsos,
total_qsos])
        wb.save(excel_path)

        # Maak het certificaat
        certificate_dir = "certificates"
        os.makedirs(certificate_dir, exist_ok=True)
        certificate_filename = os.path.join(certificate_dir,
f"{input_call}_certificate.pdf")
        create_certificate(input_call, logboeken_met_match,
certificate_filename)

    return {
        "callsign": input_call,
        "results": results,
        "total_qsos": total_qsos,
        "logbooks_with_qsos": logbooks_with_qsos,
        "qualified": qualified
    }

def create_certificate(callsign, matched_logbooks,
output_path="callsign_certificate.pdf"):
    print(callsign)
    print(matched_logbooks)
    background_path = os.path.join("static", "certificate.jpg")
    if not os.path.exists(background_path):
        print("❌ Achtergrondbestand ontbreekt:", background_path)
        return

    background = Image.open(background_path)
    width, height = A4

    c = canvas.Canvas(output_path, pagesize=A4)
    c.drawImage(ImageReader(background), 0, 0, width=width, height=height)

    font_name = "Helvetica-Bold"
    font_size = 64
    text = f"{callsign}"
    x = width / 2
    y = height - 180

    c.setFont(font_name, font_size)
    c.setFillColorRGB(1, 1, 1)
    for dx in [-1, 0, 1]:
        for dy in [-1, 0, 1]:
            if dx != 0 or dy != 0:
                c.drawCentredString(x + dx, y + dy, text)
```

```
    c.setFillColorRGB(0, 0, 0)
    c.drawCentredString(x, y, text)

    c.setFont("Helvetica-BoldOblique", 14)
    y = height - 240
    c.drawString(100, y, f"The callsign {callsign} was registered in the
following logbooks:")

    y -= 36
    for logbook in matched_logbooks:
        c.drawString(250, y, f"- {logbook}")
        y -= 16

    c.drawString(400, 50, f"Created on: {datetime.today().strftime('%Y-%m-
%d')}")
    c.save()
```

Attachment G: Results of check

- results.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Award Results for {{ data.callsign }}</title>
    <style>
        html, body {
            height: 100%;
            margin: 0;
            padding: 0;
        }

        body {
            background-image: url('{{ url_for("static", filename="result.jpg")
}}');
            background-size: cover;
            background-position: center;
            background-repeat: no-repeat;
            min-height: 100vh;
            color: white;
            font-family: sans-serif;
        }

        h1, h2, p, li {
            text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.7);
        }

        ul {
            list-style-type: none;
            padding-left: 0;
        }

        li {
            margin-bottom: 4px;
        }

        body > * {
            max-width: 800px;
            margin: 0 auto;
            padding: 1em;
        }

        .certificate-button {
            display: inline-block;
            background-color: #4CAF50;
            color: white;
            padding: 12px 24px;
            text-align: center;
            font-size: 16px;
            border: none;
            border-radius: 8px;
            text-decoration: none;
            margin-top: 20px;
            box-shadow: 1px 1px 4px rgba(0,0,0,0.3);
        }
    </style>
```

```
</head>
<body>
    <h1>Results for {{ data.callsign }}</h1>

    <ul>
    {% for logbook, count in data.results.items() %}
        <li>{{ logbook }}: {{ count }}</li>
    {% endfor %}
    </ul>

    <p><strong>Total number of QSO’s:</strong> {{ data.total_qsos }}</p>
    <p><strong>Number of logbook registrations:</strong> {{
data.logbooks_with_qsos }}</p>

    {% if data.qualified %}
        <h2>⏳ Congratulations! Your certificate is ready for you.</h2>
        <div style="text-align: center;">
            <a class="certificate-button" href="{{
url_for('download_certificate', callsign=data.callsign) }}">
                CERTIFICATE
            </a>
        </div>

    {% else %}
        <h2>😵 Unfortunately, inadequate number of QSO's for the award.</h2>
        <p>At least 1 registration is required.</p>
    {% endif %}
</body>
</html>
```

SPECIAL AWARD VALIDATION SYSTEM
PH##XMAS-PH##HNY

Callsign

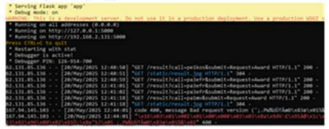
Entry Page

https://award.pa3efr.nl
@Tandata

via flaks.award.pa3efr.nl

Validation/PDF Generation

- *FLASK Server*
- *Caddy (reverse proxy)*
- *ADIF regular download*
- *Callsign_check (Python script) local ADIFs @homeQTH*



- *Generation of Certificate if positive Validation*
- *Additional info to award_winners.xlsx*

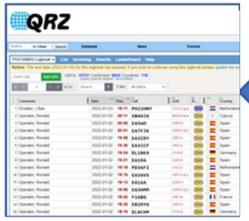
Verdict

Result Page

https://flaks.award.pa3efr.nl
@homeQTH

Download {callsign}_Certificate.pdf

Validation



Remote logbook download
@QRZ.com

Alternative page until activation