ДОДАТОК А

main.py

```
import PySimpleGUI as sg
import tensorflow as tf
import os
import random

from imageai.Detection.Custom import CustomObjectDetection
from wincapture import WindowCapture
from components import layout_all
# from speech import recognition_speech
from datetime import datetime
from dashboard import *
from utils import (
    resize_image,
    init_config,
    output_stream,
    get_titles,
    get_file_path,
    get_perm
)


os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
os.environ['TF_XLA_FLAGS'] = '--tf_xla_enable_xla_devices'

tf.config.experimental.enable_mlir_graph_optimization()
tf.config.run_functions_eagerly(True)


def start_event(window):
    global STREAMING, DETECTING, CHECKED, DETECTOR_READY
    STREAMING = True
    DETECTING = not DETECTING
    CHECKED = not CHECKED


def pause_event(window):
    global STREAMING, DETECTING, DETECTOR_READY
    STREAMING = not STREAMING
    DETECTING = not DETECTING


def targetWindow_value(window):
    try:
        target_folder = CONFIG['TARGET_WINDOW']
        window['-TARGET WINDOW-'].update(target_folder)
    except:
        pass


def targetWindow_event(window, values):
    global CONFIG
    window_title = values['-WINDOW TITLES-']
    CONFIG['TARGET_WINDOW'] = window_title
    window['-TARGET WINDOW-'].update(window_title)
    window['-WINDOW TITLES-'].update(['Unused', get_titles()])
```

```
def targetFolder_value(window):
    try:
        target_folder = CONFIG['TARGET_FOLDER']
        window['-TARGET FOLDER-'].update(target_folder)
    except:
        pass


def targetFolder_event(values):
    global CONFIG
    CONFIG['TARGET_FOLDER'] = values['-TARGET FOLDER-']


def actions_value(window):
    try:
        actions = CONFIG['ACTIONS_LOGS']
        window['-ACTIONS LOGS-'].update(actions)
    except:
        pass


def speech_value(window):
    try:
        speech = CONFIG['SPEECH_LOGS']
        window['-SPEECH LOGS-'].update(speech)
    except:
        pass


def activate_game_bot(values):
    if 'R2' not in values['-TARGET WINDOW-'] or '??' not in values['-TARGET WINDOW-']:
        sg.Popup('WARNING!\ngame is not ready',
                 title='Activate ERROR')


def streaming_event(window, values, detector):
    global DETECTIONS
    try:
        wincap = WindowCapture(values['-TARGET WINDOW-'])
        stream = wincap.get_screenshot()
    except:
        return
    DETECTIONS, stream_data = output_stream(
        stream, CONFIG['WINDOW_SIZE'], detector, detecting=DETECTING)
    window['-VIDEO STREAM-'].update(data=stream_data)


def actions_logs_event(window, values):
    global DETECTOR_COUNT
    if DETECTING and DETECTIONS is not None:
        from datetime import datetime
        now = datetime.now()
        time = now.strftime("%H:%M:%S")
        if DETECTIONS != []:
            DETECTOR_COUNT += 1
            window["-ACTIONS LOGS-"].update('\n\n', append=True)
            window["-ACTIONS LOGS-"].update('Event ' + str(DETECTOR_COUNT), text_color_for_value='white',
                                            background_color_for_value='blue', append=True)
            window["-ACTIONS LOGS-"].update('-'*18, append=True)
            window["-ACTIONS LOGS-"].update(
                '--==DETECTED==--', text_color_for_value='yellow', background_color_for_value='black',
append=True)
            window["-ACTIONS LOGS-"].update('-'*18, append=True)
```

```
            window["-ACTIONS LOGS-"].update(
                '[' + time + ']', text_color_for_value='white', background_color_for_value='blue', append=True)
            for idx, detect in enumerate(DETECTIONS):
                window["-ACTIONS LOGS-"].update('\n\n', append=True)
                window["-ACTIONS LOGS-"].update(f'{idx+1}. ', text_color_for_value='white',
                                        background_color_for_value='black', append=True)
                window["-ACTIONS LOGS-"].update('MONSTER => ' + detect['name'],
                                        text_color_for_value='black', background_color_for_value='yellow', append=True)
                window["-ACTIONS LOGS-"].update('\t', append=True)
                window["-ACTIONS LOGS-"].update('PROBABILITY => ' + str(round(detect['percentage_probability'],
3)),
                                        text_color_for_value='black', background_color_for_value='yellow', append=True)
                window["-ACTIONS LOGS-"].update('\n', append=True)
                window["-ACTIONS LOGS-"].update('POSITION => x = ' + str(detect['box_points'][0]) + ' y = ' +
str(detect['box_points'][1]),
                                        text_color_for_value='black', background_color_for_value='yellow', append=True)


def speech_window():
    window = sg.Window('Speech Recognition', layout=[
                    [sg.Button('Hello World')]])
    while True:
        event, values = window.read()

        if event == sg.WIN_CLOSED:
            break
    window.close()


def speeching_logs_event(window, values):
    global DETECTOR_COUNT
    if DETECTING and DETECTIONS is not None:
        now = datetime.now()
        time = now.strftime("%H:%M:%S")
        commands = random.choices(
            SLAVE_MESSAGE['commands'], weights=SLAVE_MESSAGE['weights'], k=random.randint(1, 3))
        # time = '-===  DETECTED  ===-'
        if DETECTIONS != []:
            for command in commands:
                window["-SPEECH LOGS-"].update('\n', append=True)
                window["-SPEECH LOGS-"].update(command,
                                        text_color_for_value='black', background_color_for_value='yellow', append=True)


def clear_event(window):
    global CONFIG, FIRST_LOAD
    FIRST_LOAD = False
    window['-START-'].update(disabled=True)
    window['-TARGET FOLDER-'].update('')
    window['-TARGET WINDOW-'].update('')
    window['-VIDEO STREAM-'].update(
        data=resize_image(CONFIG['DISCONNECT_INFO'], CONFIG['WINDOW_SIZE']))
    window['-ACTIONS LOGS-'].update('')
    window['-SPEECH LOGS-'].update('')


def program_close():
    global CONFIG
    CONFIG['TARGET_WINDOW'] = VALUES['-TARGET WINDOW-']
    CONFIG['TARGET_FOLDER'] = VALUES['-TARGET FOLDER-']


def get_window_settings():
```

```
    layout_props = [
        get_titles(),
        CONFIG['DISCONNECT_INFO'],
        CONFIG['WINDOW_SIZE']
    ]
    main_settings = dict(
        layout=layout_all(*layout_props),
        size=(1120, 630)
    )

    icon_path = get_file_path(CONFIG['WINDOW_ICON'])
    if icon_path is not None:
        main_settings.update(icon=icon_path)

    return main_settings


def save_actions_logs(values):
    with open('actions_logs.txt', 'w') as file:
        file.write(values['-ACTIONS LOGS-'])


def check_event(window, values):
    global DETECTING, CHECKED
    target_folder = values['-TARGET FOLDER-']
    perm_file = CONFIG['PERMISSION_FILE']
    if not get_perm(perm_file, target_folder, window, values):
        sg.Popup('WARNING!\n"target" or "window" is not ready',
                 title='CHECK ERROR')
    else:
        CHECKED = True


def init_detector(folder_path):
    model_path = os.path.join(
        folder_path, "models/detection_model-ex-049--loss-0012.515.h5")
    json_path = os.path.join(folder_path, "json/detection_config.json")
    detector = CustomObjectDetection()
    detector.setModelTypeAsYOLOv3()
    detector.setModelPath(model_path)
    detector.setJsonPath(json_path)
    detector.loadModel()
    return detector


def psutility(window, net_graph_in, net_graph_out, disk_graph_read, gpu_usage_graph, cpu_usage_graph,
mem_usage_graph):
    netio = psutil.net_io_counters()
    write_bytes = net_graph_out.graph_value(netio.bytes_sent)
    read_bytes = net_graph_in.graph_value(netio.bytes_recv)
    window['_NET_OUT_TXT_'].update(
        'Net out {}'.format(human_size(write_bytes)))
    window['_NET_IN_TXT_'].update(
        'Net In {}'.format(human_size(read_bytes)))
    # ----- Disk Graphs -----
    diskio = psutil.disk_io_counters()
    read_bytes = disk_graph_read.graph_value(diskio.read_bytes)
    window['_DISK_READ_TXT_'].update(
        'Disk Read {}'.format(human_size(read_bytes)))
    # ----- GPU Graph -----
    gpu = GPUtil.getGPUs()[-1].load*100
    gpu_usage_graph.graph_percentage_abs(gpu)
    window['_GPU_TXT_'].update('{0:2.0f}% GPU Used'.format(gpu))
```

```
    # ----- CPU Graph -----
    cpu = psutil.cpu_percent(0)
    cpu_usage_graph.graph_percentage_abs(cpu)
    window['_CPU_TXT_'].update('{0:2.0f}% CPU Used'.format(cpu))
    # ----- Memory Graph -----
    mem_used = psutil.virtual_memory().percent
    mem_usage_graph.graph_percentage_abs(mem_used)
    window['_MEM_TXT_'].update('{}% Memory Used'.format(mem_used))


CONFIG = init_config()
STREAMING = True
DETECTING = False
FIRST_LOAD = True
VALUES = None
CHECKED = False
DETECTIONS = None
DETECTOR_COUNT = 0
DETECTOR_READY = True


def main():
    global VALUES, DETECTOR_READY
    detector = None
    main_settings = get_window_settings()
    window = sg.Window('AI Bot', **main_settings)
    timeout = int(1000/CONFIG['FPS'])

    netio = psutil.net_io_counters()
    net_in = window['_NET_IN_GRAPH_']
    net_graph_in = DashGraph(net_in, netio.bytes_recv, '#23a0a0')
    net_out = window['_NET_OUT_GRAPH_']
    net_graph_out = DashGraph(net_out, netio.bytes_sent, '#56d856')

    diskio = psutil.disk_io_counters()
    disk_graph_read = DashGraph(
        window['_DISK_READ_GRAPH_'], diskio.read_bytes, '#5681d8')

    gpu_usage_graph = DashGraph(window['_GPU_GRAPH_'], 0, '#d34545')
    cpu_usage_graph = DashGraph(window['_CPU_GRAPH_'], 0, '#d34545')
    mem_usage_graph = DashGraph(window['_MEM_GRAPH_'], 0, '#BE7C29')

    while True:
        event, values = window.read(timeout=timeout)

        if event == sg.WIN_CLOSED:
            program_close()
            init_config(conf_file=CONFIG)
            break

        if FIRST_LOAD:
            if values['-TARGET FOLDER-'] == '':
                targetFolder_value(window)
            if values['-TARGET WINDOW-'] == '':
                targetWindow_value(window)
            if values['-ACTIONS LOGS-'] == '':
                actions_value(window)
            if values['-SPEECH LOGS-'] == '':
                speech_value(window)

        if values['-TARGET WINDOW-'] != '':
            window['-PAUSE-'].update(disabled=False)
        else:
```

```
            window['-PAUSE-'].update(disabled=True)

        if event == '-TARGET FOLDER-':
            targetFolder_event(values)
        if event == '-WINDOW TITLES-':
            targetWindow_event(window, values)

        if event == '-START-':
            start_event(window)
        if event == '-PAUSE-':
            pause_event(window)

        if CHECKED and DETECTOR_READY:
            detector = init_detector(values['-TARGET FOLDER-'])
            DETECTOR_READY = False

        if STREAMING:
            streaming_event(window, values, detector)
            actions_logs_event(window, values)
            speeching_logs_event(window, values)

        if event == '-CLEAR-':
            clear_event(window)

        if event == '-CHECK-':
            check_event(window, values)

        if event == '-ACTIVATE-':
            activate_game_bot(values)

        if event == '-SPEECH-':
            speech_window()

        if event == 'Save':
            save_actions_logs(values)

        psutility(window, net_graph_in, net_graph_out, disk_graph_read,
                gpu_usage_graph, cpu_usage_graph, mem_usage_graph)

        VALUES = {**values}

    window.close()


if __name__ == '__main__':
    main()
```

components.py

```
from utils import resize_image, get_file_path, create_image

import PySimpleGUI as sg
import os


def get_video_banner(image_path, image_size):
    sources = dict()
    path = get_file_path(image_path)
    if os.path.exists(path):
        sources.update(source=resize_image(image_path, image_size))
    else:
        sources.update(source=create_image(image_size))
    return sources


def layout_all(windows_list, image_path, image_size):
    window_menu = ['Unused', windows_list]
    header_column_left = [
        [
            sg.FolderBrowse(button_text='Target',
                            target='-TARGET FOLDER-', size=(7, 1), enable_events=True),
            sg.Input(size=(70, 1), enable_events=True,
                     readonly=True, key='-TARGET FOLDER-'),
            sg.ButtonMenu('Window', window_menu, size=(
                7, 1), key='-WINDOW TITLES-'),
            sg.Input(size=(30, 1), enable_events=True,
                     readonly=True, key='-TARGET WINDOW-')
        ]
    ]
    header_column_right = [
        [
            sg.Button(button_text='START', enable_events=True,
                      key='-START-', disabled=True),
            sg.Button(button_text='PAUSE', enable_events=True, key='-PAUSE-'),
            sg.Button(button_text='CLEAR', enable_events=True, key='-CLEAR-')
        ]
    ]
    header_layout = [
        [
            sg.Column(header_column_left,
                      element_justification='left', expand_x=True),
            sg.Column(header_column_right,
                      element_justification='right', expand_x=True)
        ]
    ]
    center_column_left = [
        [
            sg.Image(**get_video_banner(image_path, image_size), size=image_size, enable_events=True,
                     key='-VIDEO STREAM-')
        ]
    ]
    multiline_menu = ['', ['Save']]
    center_column_right = [
        [
            sg.Multiline(size=(55, 22), key='-ACTIONS LOGS-', autoscroll=True, disabled=True,
                         enable_events=True, right_click_menu=multiline_menu)
        ]
    ]
```

```
center_layout = [
   [
      sg.Column(center_column_left,
            element_justification='left'),
      sg.Column(center_column_right,
            element_justification='right', expand_x=True)
   ]
]
GRAPH_WIDTH, GRAPH_HEIGHT = 120, 40

def GraphColumn(name, key):
   layout = [
      [sg.Text(name, size=(18, 1), font=('Helvetica 8'), key=key+'TXT_')],
      [sg.Graph((GRAPH_WIDTH, GRAPH_HEIGHT),
            (0, 0),
            (GRAPH_WIDTH, 100),
            background_color='black',
            key=key+'GRAPH_')]]
   return sg.Col(layout, pad=(2, 2))
dash_layout = [
   [GraphColumn('Net Out', '_NET_OUT_'),
    GraphColumn('Net In', '_NET_IN_'),
    GraphColumn('Disk Read', '_DISK_READ_')],
   [GraphColumn('GPU Usage', '_GPU_'),
    GraphColumn('CPU Usage', '_CPU_'),
    GraphColumn('Memory Usage', '_MEM_')],
]
footer_column_left = [
   [
      sg.Column(dash_layout),
      sg.Column([
         [
            sg.Button(button_text='CHECK SETTINGS',
                  enable_events=True, key='-CHECK-', expand_x=True),
            sg.Button(button_text='ACTIVATE BOT',
                  key='-ACTIVATE-', expand_x=True)
         ],
         [
            sg.Button(button_text='SPEECH INTERVENE',
                  key='-SPEECH-', expand_x=True)
         ]
      ])
   ]
]
footer_column_right = [
   [
      sg.Column([
         [sg.Multiline(size=(55, 18), key='-SPEECH LOGS-', autoscroll=True, disabled=True,
                  enable_events=True)]
      ])
   ]
]
footer_layout = [
   [
      sg.Column(footer_column_left, element_justification='left',
            expand_x=True, expand_y=True),
      sg.Column(footer_column_right,
            element_justification='right', expand_x=True)
   ]
]
```

speech.py

```
import os
import sys
import json

import random
import time

import speech_recognition as sr

from pydub import AudioSegment


def convert_ogg_to_wav(file_path):
    input_file_path_splited = file_path.split('/')
    output_filename = input_file_path_splited[-1].replace('.ogg', '.wav')
    output_file_path = '/'.join(input_file_path_splited[:-1]) + \
        '/' + output_filename

    sound = AudioSegment.from_ogg(file_path)
    sound.export(output_file_path, format="wav")


def recognition(file_path, lang='uk-UA'):
    r = sr.Recognizer()
    audio_file = sr.AudioFile(file_path)

    with audio_file as source:
        r.adjust_for_ambient_noise(source)
        audio = r.record(source)
        recognited = r.recognize_google(audio, language=lang)

    return recognited


def save_extract_text(text, file_path=None):
    if file_path is None:
        file_path = os.path.join(os.getcwd(), 'Текст.txt')

    with open(file_path, 'w', encoding='utf-8') as file:
        file.write(text)


def get_file_path(filename):
    bundle_dir = getattr(
        sys, '_MEIPASS', os.path.abspath(os.path.dirname(__file__)))
    path_to_file = os.path.abspath(
        os.path.join(bundle_dir, 'static/' + filename))
    return path_to_file
```

```
def get_lang_tags():
    json_path = get_file_path('language_tags.json')
    with open(json_path, 'r') as file:
        lang_tags = json.load(file)
    return lang_tags


def recognize_speech_from_mic(recognizer, microphone):
    if not isinstance(recognizer, sr.Recognizer):
        raise TypeError("`recognizer` must be `Recognizer` instance")

    if not isinstance(microphone, sr.Microphone):
        raise TypeError("`microphone` must be `Microphone` instance")

    with microphone as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    response = {
        "success": True,
        "error": None,
        "transcription": None
    }

    try:
        response["transcription"] = recognizer.recognize_google(audio)
    except sr.RequestError:
        response["success"] = False
        response["error"] = "API unavailable"
    except sr.UnknownValueError:
        response["error"] = "Unable to recognize speech"

    return response


def recognition_speech():
    WORDS = ["vampire", "loot", "gold", "hold", "attack", "lemon"]
    NUM_GUESSES = 3
    PROMPT_LIMIT = 5

    recognizer = sr.Recognizer()
    microphone = sr.Microphone()

    # get a random word from the list
    word = random.choice(WORDS)

    instructions = (
        "I'm thinking of one of these words:\n"
        "{words}\n"
        "You have {n} tries to guess which one.\n"
    ).format(words=', '.join(WORDS), n=NUM_GUESSES)

    print(instructions)
    time.sleep(3)

    for i in range(NUM_GUESSES):
        for j in range(PROMPT_LIMIT):
            print('Guess {}. Speak!'.format(i+1))
            guess = recognize_speech_from_mic(recognizer, microphone)
            if guess["transcription"]:
```

```
        break
    if not guess["success"]:
        break
    print("I didn't catch that. What did you say?\n")

if guess["error"]:
    print("ERROR: {}".format(guess["error"]))
    break

print("You said: {}".format(guess["transcription"]))

guess_is_correct = guess["transcription"].lower() == word.lower()
user_has_more_attempts = i < NUM_GUESSES - 1

if guess_is_correct:
    print("Correct! You win!".format(word))
    break
elif user_has_more_attempts:
    print("Incorrect. Try again.\n")
else:
    print("Sorry, you lose!\nI was thinking of '{}'.".format(word))
    break
```

wincapture.py

```
import numpy as np
import win32gui
import win32ui
import win32con


class WindowCapture:

    w = 0
    h = 0
    hwnd = None
    cropped_x = 0
    cropped_y = 0
    offset_x = 0
    offset_y = 0

    def __init__(self, window_name):
        # find the handle for the window we want to capture
        self.hwnd = win32gui.FindWindow(None, window_name)
        if not self.hwnd:
            raise Exception('Window not found: {}'.format(window_name))

        window_rect = win32gui.GetWindowRect(self.hwnd)
        self.w = window_rect[2] - window_rect[0]
        self.h = window_rect[3] - window_rect[1]

        border_pixels = 8
        titlebar_pixels = 30
        self.w = self.w - (border_pixels * 2)
        self.h = self.h - titlebar_pixels - border_pixels
        self.cropped_x = border_pixels
        self.cropped_y = titlebar_pixels

        self.offset_x = window_rect[0] + self.cropped_x
        self.offset_y = window_rect[1] + self.cropped_y

    def get_screenshot(self):
        wDC = win32gui.GetWindowDC(self.hwnd)
        dcObj = win32ui.CreateDCFromHandle(wDC)
        cDC = dcObj.CreateCompatibleDC()
        dataBitMap = win32ui.CreateBitmap()
        dataBitMap.CreateCompatibleBitmap(dcObj, self.w, self.h)
        cDC.SelectObject(dataBitMap)
        cDC.BitBlt((0, 0), (self.w, self.h), dcObj,
                   (self.cropped_x, self.cropped_y), win32con.SRCCOPY)

        signedIntsArray = dataBitMap.GetBitmapBits(True)
        img = np.fromstring(signedIntsArray, dtype='uint8')
        img.shape = (self.h, self.w, 4)

        dcObj.DeleteDC()
        cDC.DeleteDC()
        win32gui.ReleaseDC(self.hwnd, wDC)
        win32gui.DeleteObject(dataBitMap.GetHandle())

        img = img[..., :3]

        img = np.ascontiguousarray(img)
```

```
    return img

def list_window_names(self):
    def winEnumHandler(hwnd, ctx):
        if win32gui.IsWindowVisible(hwnd):
            print(hex(hwnd), win32gui.GetWindowText(hwnd))
    win32gui.EnumWindows(winEnumHandler, None)

def get_screen_position(self, pos):
    return (pos[0] + self.offset_x, pos[1] + self.offset_y)
```

utils.py

```
from io import BytesIO
from PIL import Image

import cv2
import os
import base64
import json
import win32gui
import sys


DEFAULT_CONFIG = {
   "FPS": 20,
   "WINDOW_SIZE": [
      640,
      360
   ],
   "DISCONNECT_INFO": "disconnect_info.png",
   "WINDOW_ICON": "window_icon.ico"
}


def get_file_path(filename):
   bundle_dir = getattr(
      sys, '_MEIPASS', os.path.abspath(os.path.dirname(__file__)))
   path_to_file = os.path.abspath(
      os.path.join(bundle_dir, 'static/' + filename))
   return path_to_file


def init_config(conf_file=None, file_path='config.json'):
   file_path = get_file_path(file_path)

   if conf_file is None:
      mode = 'r'
   else:
      mode = 'w'

   file_creating = True
   while True:
      if os.path.exists(file_path):
         with open(file_path, mode) as file:
            if mode == 'r':
               conf_file = json.load(file)
               return conf_file
            json.dump(conf_file, file, indent=4)
         break
      if file_creating:
         with open(file_path, 'w') as file:
            json.dump(DEFAULT_CONFIG, file, indent=4)
         file_creating = False


def get_perm(filename, directory, window, values):
   if filename != '' and directory != '':
      if filename in os.listdir(directory):
         abs_path = os.path.join(directory, filename)
         with open(abs_path, 'r') as file:
            ready_check = json.load(file)
```

```
            if ready_check['READY'] and values['-TARGET WINDOW-'] == ready_check['WINDOW']:
                window['-START-'].update(disabled=False)
                return True
        window['-START-'].update(disabled=True)
        return False


def get_titles():
    WINDOW_LIST = []

    def winEnumHandler(hwnd, ctx):
        nonlocal WINDOW_LIST
        if win32gui.IsWindowVisible(hwnd):
            window_title = win32gui.GetWindowText(hwnd)
            if window_title != '':
                WINDOW_LIST.append(window_title)

    win32gui.EnumWindows(winEnumHandler, None)
    return WINDOW_LIST


def create_image(image_size):
    img = Image.new('RGB', image_size, color='black')
    buffered = BytesIO()
    img.save(buffered, format="PNG")
    img_str = base64.b64encode(buffered.getvalue())
    return img_str


def resize_image(filename, image_size):
    abs_path = get_file_path(filename)
    img = Image.open(abs_path)
    resized_img = img.resize(image_size)
    buffered = BytesIO()
    resized_img.save(buffered, format="PNG")
    img_str = base64.b64encode(buffered.getvalue())
    return img_str


def label_detecting(stream, detector):
    detections = detector.detectObjectsFromImage(
        input_image=stream,
        input_type='array',
        output_type='array',
        minimum_percentage_probability=40,
        extract_detected_objects=True,
        thread_safe=False
    )
    return detections[0], detections[1]


def output_stream(stream, image_size, detector, detecting=True):
    detections = None
    if detecting:
        stream, detections = label_detecting(stream, detector)
    resized = cv2.resize(
        stream, image_size, interpolation=cv2.INTER_AREA)
    imgbytes = cv2.imencode('.png', resized)[1].tobytes()
    return detections, imgbytes
```

dashboard.py

```
import PySimpleGUI as sg
import psutil
import GPUtil


# each individual graph size in pixels
GRAPH_WIDTH, GRAPH_HEIGHT = 120, 40
ALPHA = .7


class DashGraph(object):
    def __init__(self, graph_elem, starting_count, color):
        self.graph_current_item = 0
        self.graph_elem = graph_elem            # type:sg.Graph
        self.prev_value = starting_count
        self.max_sent = 1
        self.color = color
        self.graph_lines = []

    def graph_value(self, current_value):
        delta = current_value - self.prev_value
        self.prev_value = current_value
        self.max_sent = max(self.max_sent, delta)
        percent_sent = 100 * delta / self.max_sent
        line_id = self.graph_elem.draw_line(
            (self.graph_current_item, 0), (self.graph_current_item, percent_sent), color=self.color)
        self.graph_lines.append(line_id)
        if self.graph_current_item >= GRAPH_WIDTH:
            self.graph_elem.delete_figure(self.graph_lines.pop(0))
            self.graph_elem.move(-1, 0)
        else:
            self.graph_current_item += 1
        return delta

    def graph_percentage_abs(self, value):
        self.graph_elem.draw_line(
            (self.graph_current_item, 0), (self.graph_current_item, value), color=self.color)
        if self.graph_current_item >= GRAPH_WIDTH:
            self.graph_elem.move(-1, 0)
        else:
            self.graph_current_item += 1


def human_size(bytes, units=(' bytes', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB')):
    """ Returns a human readable string reprentation of bytes"""
    return str(bytes) + units[0] if bytes < 1024 else human_size(bytes >> 10, units[1:])
```

approx.py

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from utils import read_data


def drawing(x, y):
    plt.figure(figsize=(8, 6))
    plt.title("Human - RAM usage")
    plt.xlabel("Seconds")
    plt.ylabel("Usages")

    plt.scatter(x, y)

    legend = []
    fx = sp.linspace(x[0], x[-1] + 60, 1000)
    for d in range(1, 4):
        fp, residuals, rank, sv, rcond = sp.polyfit(x, y, d, full=True)
        f = sp.poly1d(fp)
        plt.plot(fx, f(fx), linewidth=2)
        legend.append("d=%i" % f.order)
        f2 = f - 1000
        t = fsolve(f2, x[-1])
    plt.legend(legend, loc="upper left")
    plt.grid()

    plt.show()


def main():
    data = read_data('human_usage.json')
    data_value = [data[key] for key in data]

    x = [x for x in range(len(data_value[-1]))]
    y = [y for y in data_value[2]]
    drawing(x, y)


if __name__ == '__main__':
    main()
```

draw.py

```
import matplotlib.pyplot as plt
from utils import read_data
import numpy as np


def moving_avg(x, n):
    cumsum = np.cumsum(np.insert(x, 0, 0))
    return (cumsum[n:] - cumsum[:-n]) / float(n)


def main():
    data = read_data('bot_usage.json')
    data_value = [data[key] for key in data]
    data_x = [x for x in range(len(data_value[-1]))]
    data_y = [y for y in data_value[2]]

    avg_y = moving_avg(data_y, 15)
    avg_x = [i for i in range(len(avg_y))]
    draw(data_x, data_y, avg_x, avg_y)
    avg_usage = round(sum(data_y)/len(data_y), 1)
    print(f'BOT RAM: {avg_usage}%')


def draw(data_x, data_y, avg_x, avg_y):
    plt.figure()
    plt.plot(data_x, data_y, linewidth=1, label='normal')
    plt.plot(avg_x, avg_y, linewidth=4, label='average')
    plt.xlabel('Seconds')
    plt.ylabel('Usages')
    plt.title('Bot - CPU usage')
    plt.legend()
    plt.grid()
    plt.show()


if __name__ == '__main__':
    main()
```

monitor.py

```
import time
import numpy as np

from utils import psutility, save_data


DATA = {'CPU': [], 'GPU': [], 'RAM': []}


def write_data(state):
    global DATA

    for key, value in zip(DATA.keys(), state):
        DATA[key].append(round(value, 1))


def main(start=0, end=10, delay=1):
    step = delay/60
    for x in np.arange(start, end, step):
        state = psutility()
        write_data(state)
        save_data(DATA)
        time.sleep(delay)


if __name__ == '__main__':
    main(0, 5, 1)
```

utils.py (analysis)

```
import psutil
import GPUtil
import json


def psutility():
    cpu = psutil.cpu_percent(0)
    gpu = GPUtil.getGPUs()[-1].load*100
    ram = psutil.virtual_memory().percent

    return cpu, gpu, ram


def read_data(filename):
    with open(filename, 'r') as f:
        data = json.load(f)
    return data


def save_data(state):
    with open('monitor_usage.json', 'w') as f:
        json.dump(state, f, indent=4)
```

convert_exe.sh

```
pyinstaller \
    --onefile --windowed \
    --add-data "static/config.json;/static/." --add-data "static/disconnect_info.png;/static/." --add-data "static/window_icon.ico;/static/." \
    --hidden-import="utils" --hidden-import="dashboard" --hidden-import="windowcapture" --hidden-import="components" \
    --name "AI Bot" \
    main.py
```

requirements.txt

absl-py==0.15.0
altgraph==0.17.2
astunparse==1.6.3
cachetools==5.2.0
certifi==2022.6.15.1
charset-normalizer==2.1.1
cycler==0.11.0
flatbuffers==1.12
future==0.18.2
gast==0.3.3
google-auth==2.11.0
google-auth-oauthlib==0.4.6
google-pasta==0.2.0
grpcio==1.32.0
h5py==2.10.0
idna==3.3
imageai==2.1.6
importlib-metadata==4.12.0
Keras==2.4.3
Keras-Preprocessing==1.1.2
keras-resnet==0.2.0
kiwisolver==1.4.4
Markdown==3.4.1
MarkupSafe==2.1.1
matplotlib==3.3.2
numpy==1.19.3
oauthlib==3.2.1
opencv-python==4.6.0.66
opt-einsum==3.3.0
pefile==2022.5.30
Pillow==7.0.0
protobuf==3.9.2
psutil==5.9.2
pyasn1==0.4.8
pyasn1-modules==0.2.8
pyinstaller==5.3
pyinstaller-hooks-contrib==2022.10
pyparsing==3.0.9
PySimpleGUI==4.60.3
python-dateutil==2.8.2
pywin32==304
pywin32-ctypes==0.2.0
PyYAML==6.0
requests==2.28.1
requests-oauthlib==1.3.1
rsa==4.9
scipy
six==1.15.0

tensorboard==2.10.0
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.1
tensorflow==2.4.0
tensorflow-estimator==2.4.0
termcolor==1.1.0
typing-extensions==3.7.4.3
urllib3==1.26.12
Werkzeug==2.2.2
wrapt==1.12.1
zipp==3.8.1

ДОДАТОК Б

Репозиторій:

https://github.com/Wizak/BIgProject.git

Файли ШІ:

https://drive.google.com/drive/folders/14WRtk8WBhtnuTLLVf_2mkZNk5lX2AbS5?usp=sharing

Відеодемонстрація:

https://youtu.be/xeejWRi8t00

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

1. Аналіз потенційних небезпек

Основними потенційними небезпеками при проведенні робіт в офісі є такі:

- небезпека ураження електричним струмом, внаслідок недотримання правил електробезпеки або виходу з ладу електроприладів;

- порушення роботи кістково-м’язового апарату внаслідок тривалих статичних навантажень при роботі з ПК;

- нервово-психічні перевантаження внаслідок тривалих статичних навантажень при роботі з ПК:

- нервово-психічні перевантаження внаслідок постійного контакту з клієнтами, колегами по роботі, керівництвом при вирішенні робочих питань, які можуть носити конфліктний характер і призвести до емоційного дискомфорту, внутрішнього роздратування, емоційної нестабільності та захворювань нервової системи;

- незадовільні ергономічні характеристики робочого місця внаслідокнераціонального планування робочого місця, що може призвести до механічних травм, уражень електричним струмом та порушень кістково- м’язового апарату;

- негативний вплив недостатнього освітлення робочої зони на зір та продуктивність роботи працюючого, внаслідок несправності освітлювальних приладів або неправильного проектування освітлювальної системи;

- негативний вплив незадовільних параметрів повітряного середовища робочої зони на здоров’я працюючого, внаслідок неправильного проектування системи вентиляції або несправності її несправності;

- негативний вплив підвищеного рівня шуму на психоемоційний стан працюючого, який пов’язаний з використанням застарілої периферійної техніки, кондиціонерів, копіювальної техніки, освітлювальних приладів;

- небезпека загоряння у зв'язку із несправністю електричного обладнання, недотримання, або порушення правил протипожежної безпеки обслуговуючим персоналом, що може призвести до пожежі.

- неправильні дії персоналу у надзвичайних ситуаціях.

2. Заходи щодо забезпечення безпеки

Приміщення офісу, в яких перебувають співробітники галузі управління персоналом, відносяться до приміщень без підвищеної небезпеки ураження електричним струмом.

Обладнання, що використовується в цих приміщеннях є споживачем електроенергії, що живиться від змінного струму 220 В від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000В закритого виконання. За способом захисту людини від ураження електричним струмом відповідає згідно з ГОСТ 12.2.007.0-75* (2001) «ССБТ.Изделия электротехнические. Общие требования безопасности» I (стаціонарні комп'ютери,) та II (освітлювальні прилади, кондиціонери, опалювальні пристрої, ноутбуки, сканери) класу захисту.

Згідно «Правилам улаштування електроустановок» (далі «ПУЕ») виконані такі групи заходів з електробезпеки:

Конструктивні заходи забезпечують захист від випадкового дотику до струмопровідних частин за допомогою їх ізоляції та захисних оболонок.

Згідно з ГОСТ 12.1.009-76 (1999) «ССБТ. Электробезопасность. Термины и определения» у приладах II класу захисту використовується подвійна ізоляція - електрична ізоляція, що складається з робочої і додаткової ізоляції.

Так як згідно з НПАОП 40.1-1.32-01 «Правила устройстваэлектроустановок. Электрооборудование специальных установок» офісні приміщення у більшості своїй відносяться до класу пожеженебезпечної зони П-IIа (приміщення, в яких містяться тверді горючі речовини), тому передбачений ступінь захисту ізоляції обладнання IP44.

Схемно-конструктивні заходи.

Призначені для забезпечення захисту від ураження електричнимструмом при дотику до металевих оболонок, які можуть опинитися під напругою в результаті аварій.

Згідно з ГОСТ 12.1.030-81 (2001) «ССБТ. Электробезопасность. Защитное заземление, зануление» у приміщеннях галузі управління персоналом влаштовується занулення.

Організаційні заходи.

Експлуатація електроустановок і електроустаткування проводиться відповідно до НПАОП 40.1-1.01-97 «Правила безпечної експлуатаціїелектроустановок» (далі «ПБЕЕ») та НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» (далі «ПБЕЕС»).

Для запобігання статистичного навантаження при користуванні ПК рекомендовано використовувати перерви в роботі 10 хв. через кожні дві години. Синдром зап’ястного каналу, або тунельний синдром зап’ястя, який може бути наслідком хронічної травми, трапляється у людей внаслідок тривалої роботи з мишею: постійні напруга і здавлювання приводить до мікротравм, здавлювання нерва прилеглими оточуючими тканинами, через що виникає набряк.

Щоб тунельний синдром вас не турбував, потрібно дотримуватися кількох правил організації робочого місця:

- оптимальна висота клавіатури від підлоги – 65-75 см;
- наявність ергономічних і зручних особисто для вас миші і клавіатури;
- можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2° до 15°);
- наявність у клавіатури підставки для рук;
- наявність килимка для миші з захистом від тунельного синдрому (спеціальний виступ забезпечує правильне положення кисті);
- наявність стільця або крісла з підлокітниками;

При роботі з мишкою і клавіатурою також слід дотримуватися певних

правила. Коли ви набираєте текст, рука повинна бути зігнута в лікті підпрямим кутом (90°), а при роботі з мишкою стежте, щоб кисть була прямою ілежала на столі якнайдалі від краю. До речі, час роботи з комп'ютером слід обмежити до дійсно необхідного.

Щоб попередити тунельний синдром потрібно робити спеціальні вправи для кистей – чим частіше, тим краще. Ці вправи допоможуть поліпшити кровообігу в м'язах і розтягнути їх. Комплекс вправ потрібноповторювати приблизно кожні 45 хвилин, тривалість однієї вправи – 1-2 хв.

Нервове напруження впливає на серцево-судинну систему, збільшуючи артеріальний тиск і частоту пульсу, а також на терморегуляцію організму та емоційні стани працівника. Особливу роль у запобіганні втомі працівників відіграють професійний відбір, організація робочого місця, правильне робочеположення, ритм роботи, раціоналізація трудового процесу, використанняемоційних стимулів, впровадження раціональних режимів праці і відпочинку тощо. Боротьба зі втомою, в першу чергу, зводиться до покращення санітарно-гігієнічних умов виробничого середовища (ліквідація забруднення повітря, шуму, вібрації, нормалізація мікроклімату, раціональне освітлення тощо).

Крім того, для профілактики втоми працівників застосовуються специфічні методи, до яких можна віднести засоби відновлення функціонального стану зорового та опорно-рухового апарату, зменшення гіподинамії, підсилення мозкового кровообігу, оптимізацію розумової діяльності. Загальні ергономічні вимоги встановлено ДСТУ ISO 9241-1:2003 «Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення». Організація робочого місця передбачає: правильне розміщення робочого місця у виробничому приміщенні; вибір ергономічно обґрунтованого робочого положення, виробничих меблів з урахуванням антропометричних характеристик людини; раціональне компонуванняобладнання на робочих місцях; врахування характеру та особливостей трудової діяльності.

В результаті технічного переоснащення, що базується на впровадженні інформаційних технологій у приміщенні, що знаходиться на другому поверсі

виробничого корпусу заплановано встановити комп'ютери. Визначимо скільки комп'ютеризованих робочих місць, оснащених відеодисплейними терміналами (ВДТ) можна встановити у даному приміщенні і як їх розташувати відповідно до встановлених норм та правил з охорони праці. Розміри приміщення: довжина $a = 5{,}1$ *м,* ширина $b = 4{,}1$ *м,* висота $h = 2{,}5$ *м.*

Перш за все необхідно проаналізувати чи підходить дане приміщення для того, щоб розмістити в ньому комп'ютеризовані робочі місця.

Відповідно до НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» є неприпустимим розташування приміщень, призначених для роботи з ВДТ у підвалах та цокольних поверхах. Також забороняється розташування вибухонебезпечних приміщень категорії А і Б (НАПБ Б.03.002-2007 «Нормы определения категорий помещений, зданий и наружных установок по взрывопожарной и пожарной безопасности») та виробництв з мокрими технологічними процесами поряд з приміщенням, де розташовуються ЕОМ (ПЕОМ), а також над такими приміщеннями, або під ними. Окрім того, виробничі приміщення для роботи з ВДТ не повинні межувати з приміщеннями, у яких рівень шумуі вібрації перевищує допустимі значення.

Вибране приміщення відповідає вищезазначеним вимогам, тому переходимо до наступного етапу, а саме, до визначення кількості комп'ютеризованих робочих місць, що можна розмістити в даному приміщенні. Оскільки площа приміщення становить $S_{ПР} = 20{,}91$ $м^2$, а площа, на якій розташовується одне робоче місце з ВДТ, відповідно до вимог ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»повинна становити не менше 6,0 $м^2$, то в даному приміщенні можна розмістити щонайбільше три комп'ютеризованих робочих місць. Такоїкількості достатньо для технічного переоснащення виробництва.

Перевіримо, чи відповідає це число нормативу щодо мінімального об'єму приміщення на одне робоче місце з ВДТ, відповідно до вимог ДСанПіН

3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»*($V_{P.M.Min}$ = 20 $м^3$*). Об'єм приміщення становить 52,275 $м^3$, а об'єм, що припадає на одне комп'ютеризоване робоче місце - $V_{P.M.}$ = 17,425 $м^3$. Таким чином норматив щодо об'єму приміщення на одне робоче місце з ВДТ не виконується. Тому розміщаємо в даному приміщенні щонайбільше два комп'ютеризованих робочих місць.

Планування розміщення комп'ютеризованих робочих місць у приміщенні проводимо із врахуванням наступних вимог:

- робочі місця з ВДТ розміщуються на відстані не менше 1 м від стіни зі світловими прорізами;

- відстань між бічними поверхнями ВДТ має бути не менше за 1,2 м;

- відстань між тильною поверхнею одного ВДТ та екраном іншого не повинна бути меншою за 2,5 м;

- прохід між рядами робочих місць має бути не меншим 1 м.

Необхідно також врахувати розміри меблів на комп'ютеризованих робочих місцях, зокрема робочого столу. Відповідно до НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» рекомендовані розміри столу для робочого місця з ВДТ становлять: висота – 725 *мм,* ширина – 600-1400 *мм,* глибина – 800-1000 *мм.* Приймаємо, що робочий стіл має такі розміри: ширина – 1200 *мм,* глибина – 800 *мм.*

Найкраще розмістити комп'ютеризовані робочі місця рядами вздовж стіни з вікнами. Це дасть змогу унеможливити дзеркальне відбиття на екрані ВДТ джерел природного світла (вікон) та потрапляння останніх у поле зору операторів, що погіршує умови їх зорової роботи.

На рис. 4.1 наведено план приміщення з комп'ютеризованими робочими місцями.

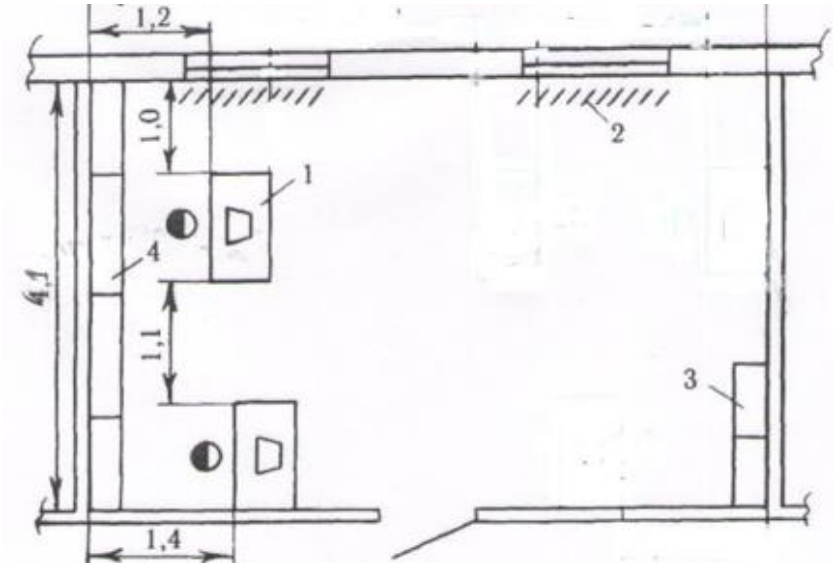

Рис. 2.1. План приміщення з комп'ютеризованими робочими місцями:

1 - комп'ютеризоване робоче місце з ВДТ; 2 - сонцезахисні жалюзі; 3 - шафи для зберігання дискет та програмного забезпечення; 4 - шафи длязберігання документації та фахової літератури.

3. Заходи щодо виробничої санітарії та гігієни праці

Згідно ДБН В.2.5-28-2006 «Інженерне обладнання будинків і споруд. Природне і штучне освітлення» в офісних приміщеннях використовується природне та штучне освітлення. Природне освітлення здійснено через світлові прорізи, які забезпечують коефіцієнт природної освітленості (КПО) не нижче 1,5%. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, використовуються сонцезахисні пристрої, на вікнах встановлені жалюзі або штори. Штучне освітлення в приміщенні здійснено системою загального рівномірного освітлення. Значення освітленості на поверхні робочого столу взоні розміщення документів становить 300-500 лк. Як джерела штучного освітлення в приміщенні застосовуються люмінесцентні лампи типу ЛБ.

Показники мікроклімату в офісних приміщеннях відповідають встановленим санітарно-гігієнічним вимогам ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень», ГОСТ 12.1.005-88 (1991)

«ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны» і ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень».

Роботи в приміщеннях обслуговування туристів, належать до категорії Іб - легка робота, тому встановлені наступні оптимальні значення параметрів мікроклімату:

- у холодний період року: температура 21-23°С; відносна вологість: 40-60%; швидкість переміщення повітря: 0,1 м/с;

- у теплий період року: температура 22-24°С; відносна вологість: 40-60%; швидкість переміщення повітря: 0,2 м/с.

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщеня з ПК нормовані згідно ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» істановлять: n+=1500-30000 (шт. на 1см^3); n- = 3000-5000 (шт. на 1см^3). Підтримка оптимального рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів.

Дотримання вимог цих документів досягається оснащенням приміщень і транспортних засобів пристроями кондиціонування і вентиляції, дезодорації повітря, опалювання згідно вимог ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування».

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»та ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку таінфразвуку».

Зниження рівня шуму в приміщенні здійснено за допомогою:

- використання більш сучасного обладнання;

- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;

- переведення жорсткого диска в режим сну (Standby), якщо комп'ютер

не працює протягом визначеного часу;

- використання блоків живлення ПК з вентиляторами на гумових підвісках.

4. Заходи з пожежної безпеки

Закон України «Про пожежну безпеку» визначає загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулює відносини державних органів, юридичних і фізичних осіб уцій галузі незалежно від виду їх діяльності та форм власності.

Пожежна безпека – стан об'єкта, при якому з регламентованою ймовірністю виключається можливість виникнення та розвиток пожежі і впливу на людей її небезпечних факторів, а також забезпечується захистматеріальних цінностей.

Для забезпечення пожежної безпеки в установах проводять пожежну профілактику, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння. До них відносяться: вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, ящики з піском, пожежні відра, совкові лопати, ломи, сокири тощо), системи автоматичного пожежогасіння.

Первинні засоби пожежогасіння, в залежності від категорії приміщень, можуть розташовуватись як окремо, так і в складі пожежних щитів.

Залежно від агрегатного стану й особливостей горіння різних горючих речовин і матеріалів пожежі за ДБН В.1.1.7-2002 «Пожежна безпека об'єктів будівництва» поділяються на відповідні класи.

В офісному приміщенні знаходиться дерев`яна мебель, електронна апаратура, бумажні носії інформації.

Клас пожежі у офісному приміщенні (згідно із ДБН В.1.1.7-2002

«Захист від пожежі. Пожежна безпека об’єктів будівництва») – пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластмаси, папір) – визначається як клас А. Категорія приміщення (згідно із НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок завибухопожежною та пожежною небезпекою») – визначається як категорії Д.

Визначення типу та розрахунок кількості первинних засобів пожежегасіння (згідно із ДБН В.1.1.7-2002 «Захист від пожежі. Пожежна безпека об’єктів будівництва») – для адміністративного приміщення площею 20,91 м^2 слід застосовувати два порошкових вогнегасниками типу ВП–5 (НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». ДБН В.1.1.7-2002 «Пожежна безпека об’єктів будівництва»). Крім цього адміністративні приміщення повинні бути обладнані автоматичними пожежними сповіщувачами, що реагують на підвищення температури, дим,полум’я. Наприклад, сповіщувачі моделей ДТЛ, ІТМ.

5. Заходи безпеки у надзвичайних ситуаціях

Ударною хвилею називається область різкого стиску середовища, що поширюється у вигляді сферичного шару від місця вибуху з надзвуковою швидкістю. Ударні хвилі класифікуються в залежності від середовища поширення. Ударна хвиля в повітрі виникає за рахунок передачі стиснення і розширення шарів повітря. Зі збільшенням відстані від місця вибуху хвиля слабшає і перетворюється на звичайну акустичну. Хвиля при проходженні через дану точку простору викликає зміни в тиску, що характеризуються наявністю двох фаз: стиснення та розширення. Період стиснення настає відразу і триває порівняно невеликий час в порівнянні з періодом розширення. Руйнівна дія ударної хвилі характеризують надлишковий тиск у її фронті (передній межі), тиск швидкісного напору, тривалість фазистиснення.

Ударна хвиля у воді відрізняється від повітряної значеннями своїх характеристик (великим надмірним тиском і меншим часом впливу). Ударна хвиля в ґрунті при видаленні від місця вибуху стає подібна сейсмічної хвилі. Вплив ударної хвилі на людей і тварин може призвести до отримання безпосередніх чи непрямих поразок. Воно характеризується легкими, середніми, важкими і вкрай важкими ушкодженнями та травмами.

Вражаюче дія ударної хвилі характеризується величиною надлишкового тиску. Надмірний тиск – це різниця між максимальним тискому фронті ударної хвилі і нормальним атмосферним тиском перед ним.

При надлишковому тиску 20-40 кПа незахищені люди можуть одержати легкі поразки (легкі забиті місця і контузії). Вплив ударної хвилі з надлишковим тиском 40-60 кПа призводить до поразок середньої важкості: втраті свідомості, ушкодженню органів слуху, сильним вивихів кінцівок, кровотечі з носа і вух. Важкі травми виникають при надлишковому тиску понад 60 кПа. Вкрай важкі поразки спостерігаються при надлишковому тискупонад 100 кПа.

Ударна хвиля ядерного вибуху може на значній відстані від центра вибуху завдавати поразки людям, руйнувати споруди і ушкоджувати бойову техніку. Ударна хвиля являє собою область сильного стиснення повітря, що розповсюджується з великою швидкістю у всі сторони від центра вибуху. Швидкість поширення її залежить від тиску повітря у фронті ударної хвилі; поблизу центра вибуху вона в декілька разів перевищує швидкість звуку, але із збільшенням відстані від місця вибуху різко падає

Вражаюча дія ударної хвилі на людей і руйнуючу дію на бойову техніку, інженерні споруди і матеріальні кошти передусім визначаються надмірним тиском і швидкістю рушення повітря в її фронті. Незахищенілюди можуть, крім того поранитися осколками скла, що летять з величезною швидкістю і обломками будівель, що руйнуються, падаючими деревами, а також частинами бойової техніки, що розкидаються, камінням і іншими предметами, що приводяться в рушення швидкісним натиском ударної хвилі. Найбільші непрямі поразки будуть спостерігатися в населених пунктах і в лісі; в цих випадках втрати військ

можуть виявитися більшими, ніж від безпосередньої дії ударної хвилі. Ударна хвиля здатна завдавати поразки і в закритих приміщеннях, проникаючи туди через щілини і отвори. Поразки, щонаносяться ударною хвилею, поділяються на легкі, середні, важкі і надто важкі. Легкі поразки характеризуються тимчасовим пошкодженням органів слуху, загальною легкою контузією, ударами і вивихами кінцівок. Важкі поразки характеризуються сильною контузією всього організму; при цьому можуть спостерігатися пошкодження головного мозку і органів черевної порожнини, сильна кровотеча з носа і вух, важкі переломи і вивихи кінцівок. Міра поразки ударною хвилею залежить передусім від потужності і вигляду ядерного вибуху.

Механічне вплив ударної хвилі оцінюється за ступенем руйнувань, викликаних дією хвилі (виділяються слабке, середнє, сильне і повнеруйнування). Енергетичне, промислове і комунальне обладнання в результаті дії ударної хвилі може отримати пошкодження, також оцінюються за їх тяжкості (слабкі, середні і сильні). Вплив ударної хвилі може призвести також до пошкоджень транспортних засобів, гідровузлів, лісових масивів. Як правило, шкода, яка завдається впливом ударної хвилі, дуже великий, він наноситься як здоров'ю людей, так і різним спорудам, устаткуванню і т.д.

Середнє руйнування проявляється в руйнуванні дахів і вбудованих елементів-внутрішніх перегородок, вікон, а також у виникненні тріщини у стінах, обвалення окремих ділянок горищних перекриттів і стін верхніх поверхів. Підвали зберігаються. Після розчистки і ремонту може бути використана частина приміщень нижніх поверхів. Відновлення будівель можливо при проведенні капітального ремонту.

Сильне руйнування характеризується руйнуванням несучихконструкцій і перекриттів верхніх поверхів, утворенням тріщин у стінах і деформацією перекриття нижніх поверхів. Використання приміщень стає неможливим, а ремонт і відновлення найчастіше недоцільним.

Повне руйнування. Руйнуються всі основні елементи будівлі,включаючи

і несучі конструкції. Використовувати будівлі неможливо. Підвальні приміщення при сильних та повних руйнування можуть зберігатися і після розбору завалів частково використовуватися.

Таким чином, обладнання, що використовується в цих приміщеннях є споживачем електроенергії, що живиться від змінного струму 220 В від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000 В закритого виконання. За способом захисту людини від ураження електричним струмом відповідає згідно з ГОСТ 12.2.007.0-75* (2001) «ССБТ.Изделия электротехнические. Общие требования безопасности» I (стаціонарні комп'ютери,) та II (освітлювальні прилади, кондиціонери, опалювальні пристрої, ноутбуки, сканери) класу захисту.

Внаслідок тривалих статичних навантажень при роботі з ПК може виникнути так званий синдром зап`ясного каналу. Для профілактики і лікування синдрому зап'ястного каналу слід правильно організувати власне робоче місце, якомога частіше переривати роботу і виконувати невеликий комплекс вправ для рук. Щогодини слід робити перерву, хоча б на 3-5 хвилин, піднімати кисті рук вгору і робити невелику зарядку для рук.

Особливу роль у запобіганні втомі працівників відіграють професійний відбір, організація робочого місця, правильне робоче положення, ритм роботи, раціоналізація трудового процесу, використання емоційних стимулів,впровадження раціональних режимів праці і відпочинку тощо. Боротьба зі втомою, в першу чергу, зводиться до покращення санітарно-гігієнічних умов виробничого середовища (ліквідація забруднення повітря, шуму, вібрації, нормалізація мікроклімату, раціональне освітлення тощо).

Загальні ергономічні вимоги встановлено ДСТУ ISO 9241-1:2003 «Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення». Організація робочого місця передбачає: правильне розміщення робочого місця у виробничому приміщенні; вибір ергономічно обґрунтованого робочого положення, виробничих меблів з урахуванням антропометричних ха-

рактеристик людини; раціональне компонуванняобладнання на робочих місцях; врахування характеру та особливостей трудової діяльності.

Штучне освітлення в приміщенні здійснено системою загального рівномірного освітлення. Значення освітленості на поверхні робочого столу взоні розміщення документів становить 300-500 лк. Як джерела штучного освітлення в приміщенні застосовуються люмінесцентні лампи типу ЛБ.

Показники мікроклімату в офісних приміщеннях відповідають встановленим санітарно-гігієнічним вимогам ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень», ГОСТ 12.1.005-88 (1991) «ССБТ. Общие санитарно-гигиенические требования к воздуху рабочиезоны» і ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень».

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»та ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку таінфразвуку».

Визначення типу та розрахунок кількості первинних засобів пожежегасіння згідно із ДБНВ.1.1.7-2002 «Захист від пожежі. Пожежна безпека об’єктів будівництва» - для адміністративного приміщення площею 20,91 м^2 слід застосовувати два порошкових вогнегасниками типу ВП-5 згідно НАПБ Б.03.001-2004 «Типові норми належності вогнегасників». Крім цього адміністративні приміщення повинні бути обладнані автоматичними пожежними сповіщувачами, що реагують на підвищення температури, дим, полум’я.

Ударною хвилею називається область різкого стиску середовища, що поширюється у вигляді сферичного шару від місця вибуху з надзвуковою швидкістю.