ДОДАТКИ

Додаток А

Харківський національний університет радіоелектроніки
Кафедра КІТС

Інтелектуальна система для розпізнавання номерних знаків автомобілів за допомогою технології OCR за наявності шуму

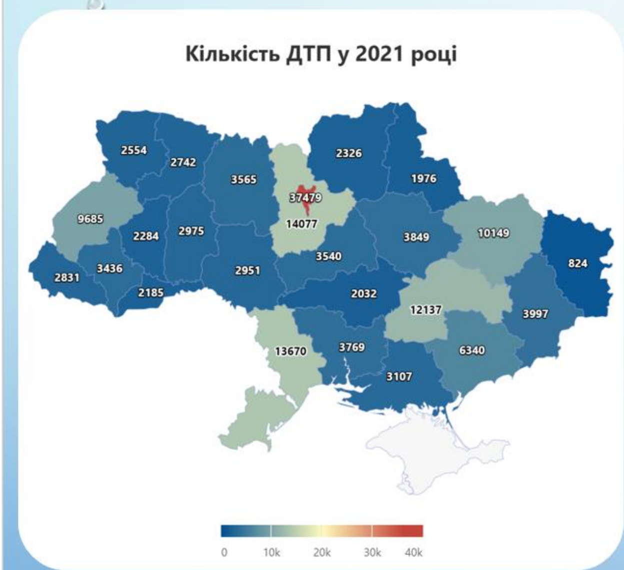
Кваліфікаційна робота
Магістерський рівень

Виконав:
студент групи КІТм-21-1
Омельченко С. О.

Керівник:
ст. викладач, к.т.н. Ілюнін О.О.

2

Актуальність роботи



3

Мета роботи

Метою кваліфікаційної роботи є створення програмного забезпечення для автоматичного розпізнавання зображень номерних знаків автомобілів за допомогою технології оптичного розпізнавання символів за наявності шуму.

Об’єктом дослідження є процеси розпізнавання зображень номерних знаків автомобілів за допомогою технології оптичного розпізнавання символів за наявності шуму.

Предметом дослідження є нейро-нечіткі методи та моделі розпізнавання зображень, образів за наявності шуму, а саме – номерних знаків автомобілів за допомогою технології оптичного розпізнавання символів за наявності шуму.

4

Комп’ютерний зір

Це напрям досліджень штучного інтелекту, який дозволяє комп'ютерам і системам витягувати значиму інформацію з цифрових зображень, відео та інших візуальних входів і вживати заходи або давати рекомендації на основі цієї інформації

Напрямки застосування:

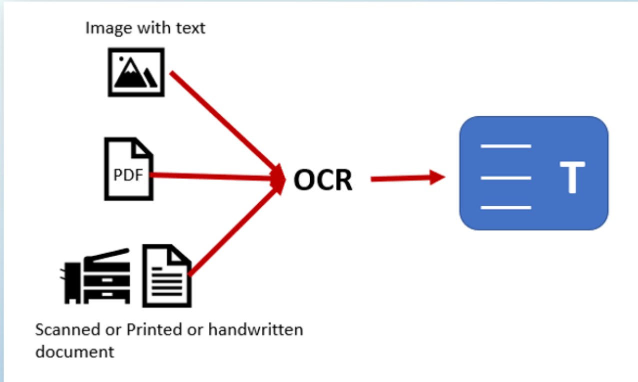
- розпізнавання образів
- виявлення об’єктів
- виявлення подій
- 3D оцінка пози
- моделювання 3D-сцен
- реконструкція сцени
- відновлення зображення
- відстеження руху
- охорона здоров’я

5

Оптичне розпізнавання символів (OCR)

Варіанти використання:

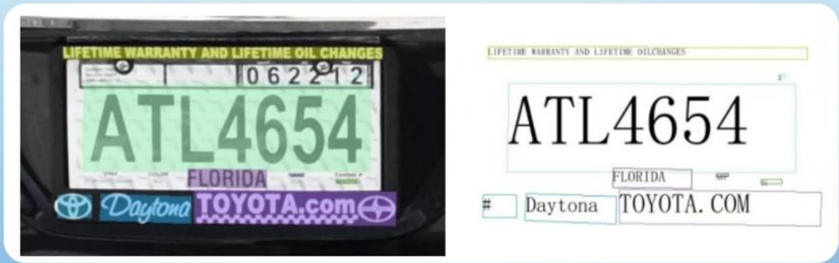
- перетворення друкованих паперових документів та документів зі сканованими зображеннями у машинозчитувані текстові документи
- розпізнавання даних паспортів для аеропортів
- розпізнавання дорожніх знаків
- вилучення контактної інформації з документів або візиток



6

Автоматичне розпізнавання номерних знаків (ANPR)

Це практика комп’ютерного зору, яка дозволяє пристроям швидко та автоматично читати номерні знаки транспортних засобів без будь-якої людської взаємодії.
ANPR використовує зображеннях номерних знаків транспортного засобу для зчитування реєстраційного номера автомобіля за допомогою технології OCR.



7

Цифровий шум зображення

Види цифрового зображення:

- бінарне зображення
- зображення в градаціях сірого
- кольорове зображення

$$w(x, y) = s(x, y) + n(x, y)$$

s(x, y) – початкове зображення
n(x, y) – шум у позиціях пікселя (x, y)
w(x, y) – спотворене зображення

Основні джерела шуму:

- слабке освітлення
- режими високої чутливості
- датчики слабкої продуктивності
- температура датчика
- щільність розміщення пікселів
- частинки пилу в сканері
- перешкоди каналу передачі
- фактори навколишнього середовища

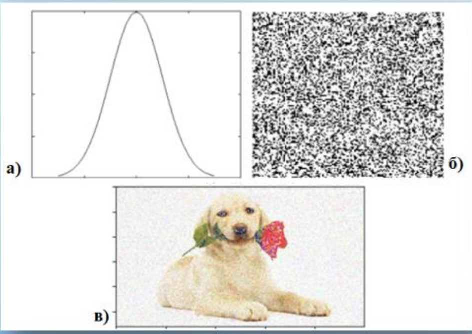
8

Типи шуму

Гаусовий шум

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

z – представляє рівень сірого
μ – середнє значення сірого
σ –стандартне відхилення

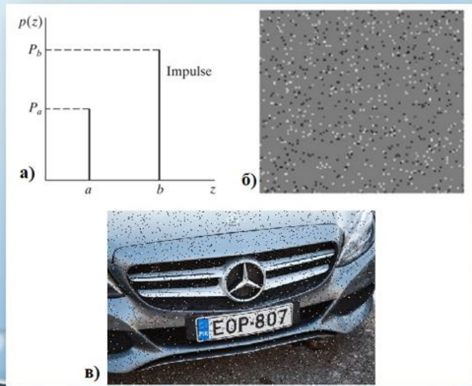


а) графік розподілу шуму
б) піксельне подання шуму
в) результат накладання шуму

Шум «солі та перцю»

$$p(z) = \begin{cases} P_a, z = a \\ P_b, z = b \\ 0 \end{cases}$$

Pa – ймовірність значення шуму «солі»
Pb – ймовірність значення шуму «перцю»



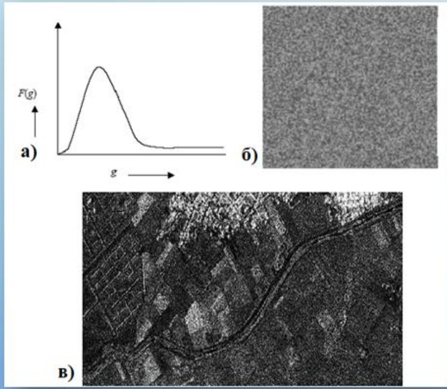
9

Типи шуму

Спекл-шум

$$F(g) = \frac{g^{a-1}}{(a-1)!\,a^{a}} e^{\frac{-g}{a}}$$

a – дисперсія
g – рівень сірого

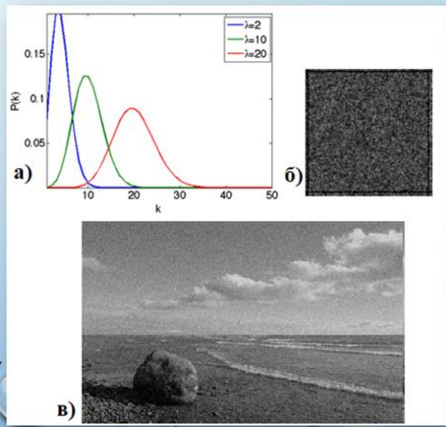


а) графік розподілу шуму
б) піксельне подання шуму
в) результат накладання шуму

Шум Пуассона

$$P(k) = \frac{\lambda^{k} e^{-\lambda}}{k!}$$

k – значення 0,1,2 …
λ – середнє число входжень в інтервалі



10

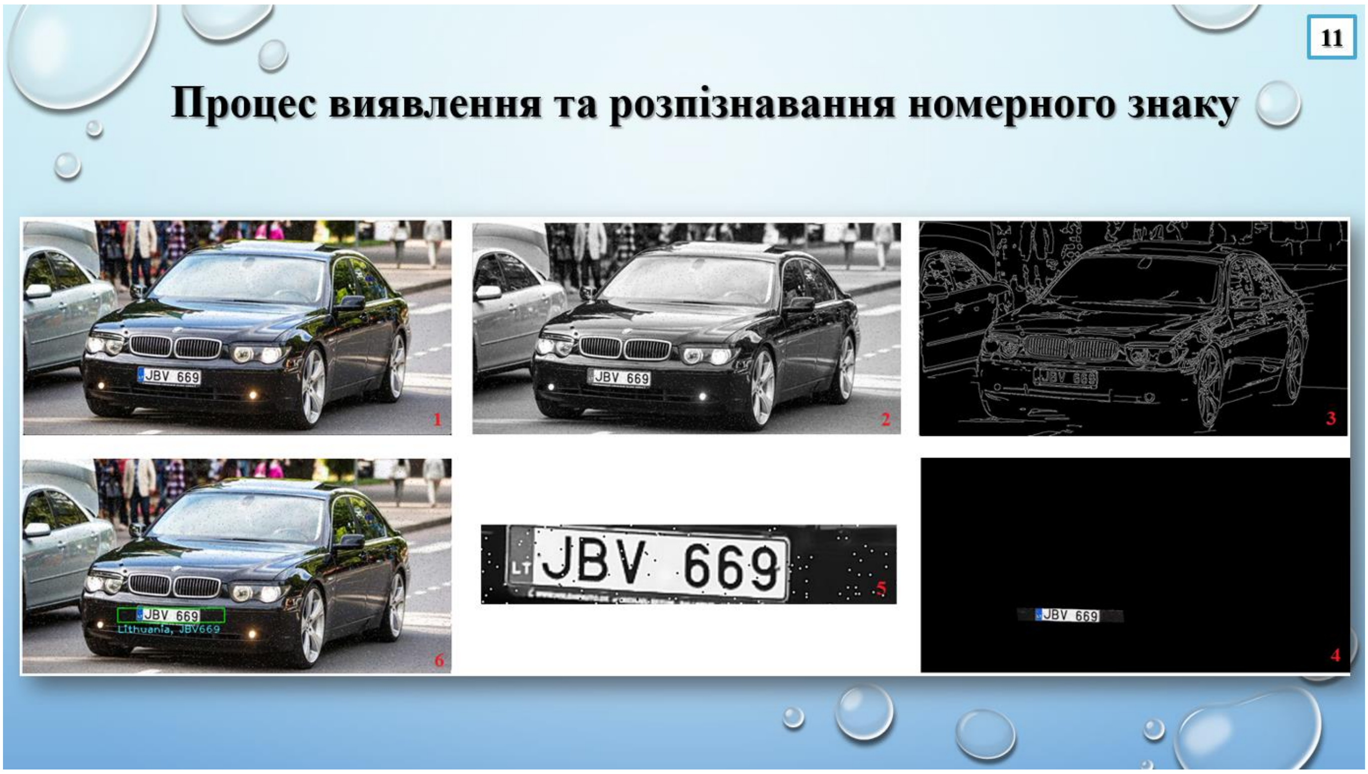
Технології розробки та інструментальні засоби

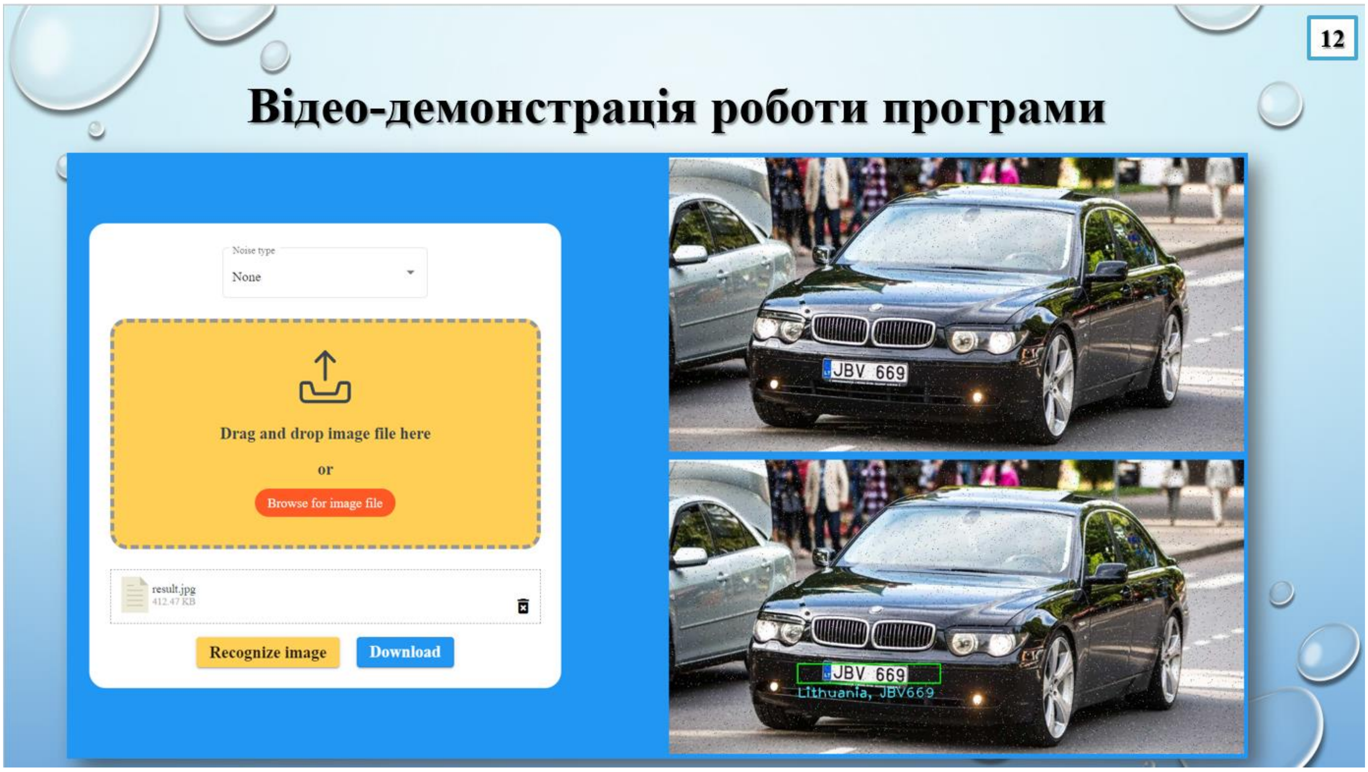
Клієнтська частина

- фреймворк Angular
- мова програмування TypeScript
- середовище розробки WebStorm

Серверна частина

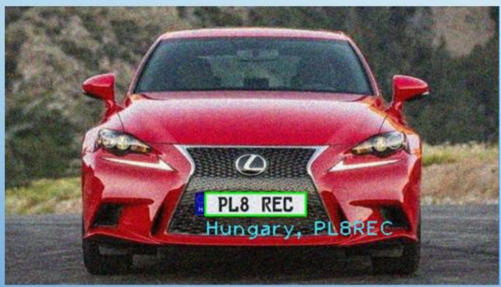
- фреймворк Flask
- бібліотеки OpenCV, NumPy, Imutils та EasyOCR
- мова програмування Python
- середовище розробки PyCharm


11
Процес виявлення та розпізнавання номерного знаку
JBV 669
1
2
3
LT JBV 669
5
Lithuania, JBV669
6
4


12
Відео-демонстрація роботи програми
Noise type
None
Drag and drop image file here
or
Browse for image file
result.jpg
412.47 KB
Recognize image
Download
JBV 669
Lithuania, JBV669

13

Приклади результатів розпізнавання номерних знаків



14

Результати експериментальних досліджень

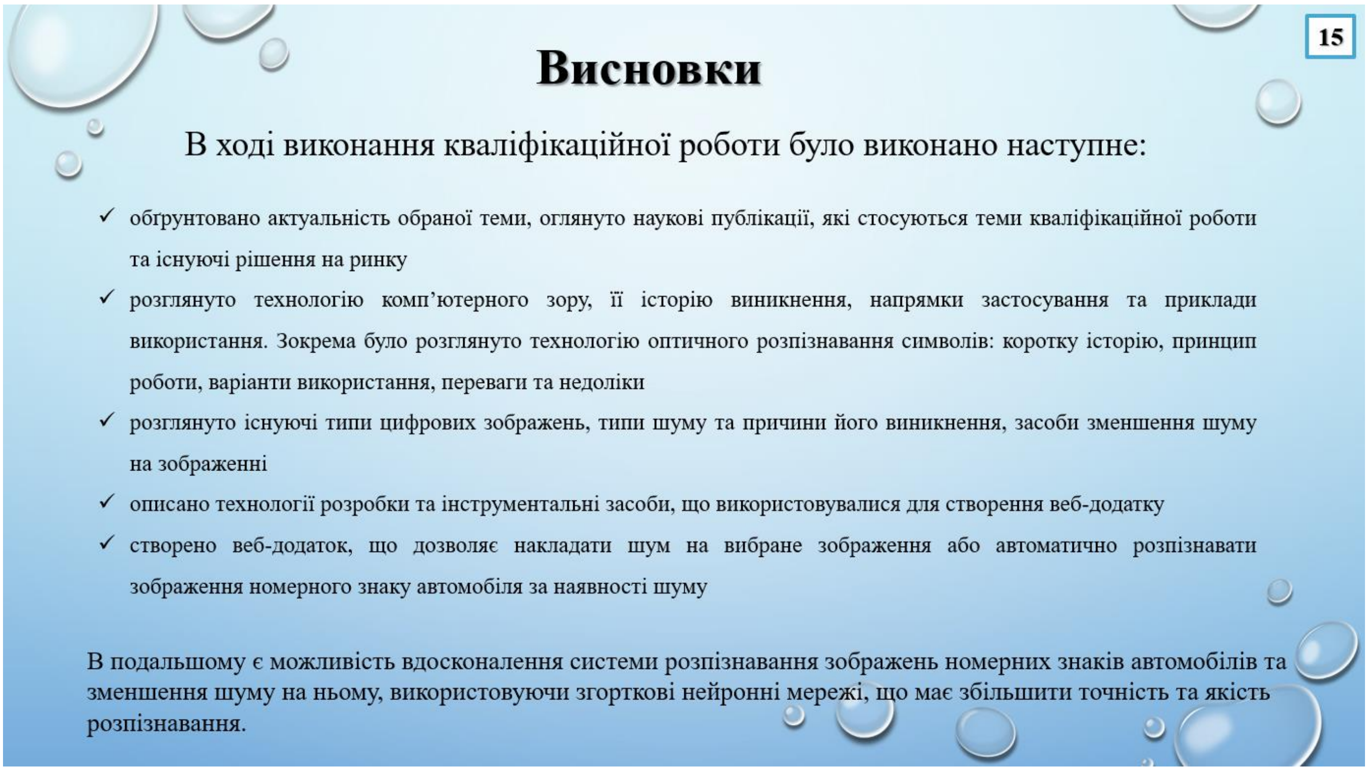
$$SR = \frac{NS_s}{TN_s} * 100$$

SR – коефіцієнт успішності роботи алгоритму
NSs – кількість успішних зразків
TNs – загальна кількість зразків зображень

Назва операції	Загальна кількість зразків	Кількість успішних зразків	Кількість невдалих спроб	Коефіцієнт успішності (%)
Вилучення та виявлення номерного знаку	50	46	4	92
Розпізнавання тексту номерного знаку	46	44	2	95,65

Графік залежності часу виконання алгоритму від розміру вхідного зображення

Середній час роботи алгоритму: ~2 – 6 секунд


15
Висновки
В ході виконання кваліфікаційної роботи було виконано наступне:
✓ обґрунтовано актуальність обраної теми, оглянуто наукові публікації, які стосуються теми кваліфікаційної роботи та існуючі рішення на ринку
✓ розглянуто технологію комп’ютерного зору, її історію виникнення, напрямки застосування та приклади використання. Зокрема було розглянуто технологію оптичного розпізнавання символів: коротку історію, принцип роботи, варіанти використання, переваги та недоліки
✓ розглянуто існуючі типи цифрових зображень, типи шуму та причини його виникнення, засоби зменшення шуму на зображенні
✓ описано технології розробки та інструментальні засоби, що використовувалися для створення веб-додатку
✓ створено веб-додаток, що дозволяє накладати шум на вибране зображення або автоматично розпізнавати зображення номерного знаку автомобіля за наявності шуму
В подальшому є можливість вдосконалення системи розпізнавання зображень номерних знаків автомобілів та зменшення шуму на ньому, використовуючи згорткові нейронні мережі, що має збільшити точність та якість розпізнавання.

Додаток Б

Лістинг програми

Б.1 Лістинг клієнтської частини програми

Код сторінки image.component.html представлено в лістингу Б.1

Лістинг Б.1 – Файл image.component.html

```
<div class="container" appDnd (fileDropped)="onFileDropped($event)">
  <input type="file" #fileDropRef id="fileDropRef" (change)="fileBrowseHandler($event)" accept="image/png, image/jpeg, image/jpg" />
  <svg xmlns="http://www.w3.org/2000/svg" width="63" height="64" viewBox="0 0 63 64">
    <g fill="#3B454F" fill-rule="nonzero">
      <path
        d="M42.656 15.135a1.953 1.953 0 0 1-1.391-.578L31.5 4.795l-9.765 9.762a1.97 1.97 0 1 1-2.785-2.785L30.106.616a1.97 1.97 0 0 1 2.785 0l11.157 11.156a1.97 1.97 0 0 1-1.392 3.363z" />
      <path
        d="M31.5 36.791a1.97 1.97 0 0 1-1.969-1.969V2.01a1.97 1.97 0 0 1 3.938 0v32.812a1.97 1.97 0 0 1-1.969 1.969z" />
      <path
        d="M55.781 63.041H7.22A7.225 7.225 0 0 1 0 55.822V41.385a4.599 4.599 0 0 1 4.594-4.594h7.234a4.567 4.567 0 0 1 4.402 3.276l2.814 9.382a.658.658 0 0 0 .628.467h23.656a.658.658 0 0 0 .628-.467l2.814-9.385a4.572 4.572 0 0 1 4.402-3.273h7.234A4.599 4.599 0 0 1 63 41.385v14.437a7.225 7.225 0 0 1-7.219 7.219zM4.594 40.729a.656.656 0 0 0-.657.656v14.437a3.286 3.286 0 0 0 3.282 3.282H55.78a3.286 3.286 0 0 0 3.282-3.282V41.385a.656.656 0 0 0-.657-.656h-7.234a.65.65 0 0 0-.628.467L47.73 50.58a4.628 4.628 0 0 1-4.402 3.274H19.672a4.567 4.567 0 0 1-4.402-3.276l-2.814-9.382a.65.65 0 0 0-.628-.467H4.594z" />
    </g>
  </svg>

  <h3>Drag and drop image file here</h3>
  <h3>or</h3>
  <label for="fileDropRef">Browse for image file</label>
</div>
<div class="files-list">
```

```
       <div  class="single-file"  *ngFor="let  file  of  files;  let  i  =
index">
         <div class="file-icon" style="width: 50px">
           <svg               version="1.1"                id="Capa_1"
xmlns="http://www.w3.org/2000/svg"
                x="0px"  y="0px"  viewBox="0  0  58  58"  style="enable-
background:new 0 0 58 58;" xml:space="preserve">
                        <polygon style="fill:#EDEADA;" points="51.5,14
37.5,0 6.5,0 6.5,58 51.5,58 " />
             <g>
                             <path style="fill:#CEC9AE;"
                     d="M16.5,23h25c0.552,0,1-0.447,1-1s-0.448-1-1-1h-
25c-0.552,0-1,0.447-1,1S15.948,23,16.5,23z" />
               <path style="fill:#CEC9AE;"
                     d="M16.5,15h10c0.552,0,1-0.447,1-1s-0.448-1-1-1h-
10c-0.552,0-1,0.447-1,1S15.948,15,16.5,15z" />
               <path style="fill:#CEC9AE;"
                     d="M41.5,29h-25c-0.552,0-1,0.447-
1,1s0.448,1,1,1h25c0.552,0,1-0.447,1-1S42.052,29,41.5,29z" />
               <path style="fill:#CEC9AE;"
                     d="M41.5,37h-25c-0.552,0-1,0.447-
1,1s0.448,1,1,1h25c0.552,0,1-0.447,1-1S42.052,37,41.5,37z" />
               <path style="fill:#CEC9AE;"
                     d="M41.5,45h-25c-0.552,0-1,0.447-
1,1s0.448,1,1,1h25c0.552,0,1-0.447,1-1S42.052,45,41.5,45z" />
                        </g>
             <polygon  style="fill:#CEC9AE;"  points="37.5,0  37.5,14
51.5,14 " />
                    </svg>
         </div>
         <div class="info">
           <h4 class="name">
             {{ file.name }}
           </h4>
           <p class="size">
             {{ formatBytes(file.size) }}
           </p>
         </div>

         <div class="delete">
           <mat-icon class="delete-icon" (click)="deleteFile(i)">
             delete_forever
           </mat-icon>
         </div>
       </div>
     </div>
```

Клас ImageComponent.ts представлено в лістингу Б.2.

Лістинг Б.2 – Клас ImageComponent.ts

```
import {Component, EventEmitter, OnInit, Output} from '@angular/core';
import {MessageService} from "../../services/message.service";

@Component({
  selector: 'app-image',
  templateUrl: './image.component.html',
  styleUrls: ['./image.component.scss']
})
export class ImageComponent implements OnInit {

  @Output() outputImageUrl = new EventEmitter<string>();
  @Output() outputSelectedImage = new EventEmitter<File>();

  files: File[] = [];

  constructor(private messageService: MessageService) {
  }

  ngOnInit(): void {
  }

  onFileDropped(file: File) {
    if (this.isValidImageExtension(file.name)) {
      this.prepareFilesList(file);
      this.displaySelectedImage(file);
    }
  }

  fileBrowseHandler(event: Event) {
    const target = event.target as HTMLInputElement;
    if (this.isValidImageExtension(target.files![0].name)) {
      this.prepareFilesList(target.files![0]);
      this.displaySelectedImage(target.files![0]);
    }
  }

  deleteFile(index: number) {
    this.files.splice(index, 1);
    this.outputImageUrl.emit(undefined)
  }

  prepareFilesList(file: File) {
    this.deleteFile(0);
    this.files.push(file);
  }

  formatBytes(bytes: number) {
```

```
    if (bytes === 0) {
      return '0 Bytes';
    }
    const k = 1024;
    const sizes = ['Bytes', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'];
    const i = Math.floor(Math.log(bytes) / Math.log(k));
    return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + ' ' + sizes[i];}
  displaySelectedImage(file: File) {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = (event: ProgressEvent<FileReader>) => {
      this.outputImageUrl.emit(event.target!.result!.toString())
    }
    this.outputSelectedImage.emit(file);
  }

  isValidImageExtension(name: string): boolean {
    const fileExtension = name.split(".")[1].toLowerCase();
    if (fileExtension != "jpg" && fileExtension != "jpeg" && fileExtension != "png") {
      this.messageService.open('Invalid input file format: only .jpeg, .jpg, and .png formats are allowed.')
      return false;
    } else {
      return true;
    }
  }
}
```

Файл image.component.scss представлено лістингом Б.3.

Лістинг Б.3 – Файл image.component.scss

```
@import "~src/_variables.scss";

.container {
  width: 450px;
  height: 200px;
  padding: 2rem;
  text-align: center;
  border-radius: 20px;
  border: dashed 5px #979797;
  position: relative;
  margin: 0 auto;
  background-color: #ffd055;

  input {
```

```
    opacity: 0;
    position: absolute;
    z-index: 2;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
  }

  label {
    color: white;
    width: 183px;
    height: 44px;
    border-radius: 21.5px;
    background-color: $red;
    padding: 8px 16px;
  }

  h3 {
    font-size: 20px;
    font-weight: 600;
    color: #38424c;
  }
}

.fileover {
  animation: shake 1s;
  animation-iteration-count: infinite;
}

.files-list {
  margin-top: 1.5rem;

  .single-file {
    display: flex;
    padding: 0.5rem;
    justify-content: space-between;
    align-items: center;
    border: dashed 1px #979797;
    margin-bottom: 1rem;

    .delete {
      display: flex;
      margin-left: 0.5rem;
      cursor: pointer;
      align-self: flex-end;
    }

    .delete-icon:hover {
      color: $red;
    }
    flex-grow: 1;
```

```
    .name {
      font-size: 14px;
      font-weight: 500;
      color: #353f4a;
      margin: 0;
    }

    .size {
      font-size: 12px;
      font-weight: 500;
      color: #a4a4a4;
      margin: 0;
      margin-bottom: 0.25rem;
    }

    .info {
      width: 100%
    }
  }
}

/* Shake animation */
@keyframes shake {
  0% {
    transform: translate(1px, 1px) rotate(0deg);
  }

  10% {
    transform: translate(-1px, -2px) rotate(-1deg);
  }

  20% {
    transform: translate(-3px, 0px) rotate(1deg);
  }

  30% {
    transform: translate(3px, 2px) rotate(0deg);
  }

  40% {
    transform: translate(1px, -1px) rotate(1deg);
  }

  50% {
    transform: translate(-1px, 2px) rotate(-1deg);
  }

  60% {
    transform: translate(-3px, 1px) rotate(0deg);
  }

  70% {
    transform: translate(3px, 1px) rotate(-1deg);
```

```
  }

  80% {
    transform: translate(-1px, -1px) rotate(1deg);
  }

  90% {
    transform: translate(1px, 2px) rotate(0deg);
  }

  100% {
    transform: translate(1px, -2px) rotate(-1deg);
  }
}
```

Файл main.component.html представлено в лістингу Б.4.

Лістинг Б.4 – Файл main.component.html

```
<div class="main-container">
  <mat-progress-bar *ngIf="loading" class="progress-bar" mode="indeterminate" color="accent"></mat-progress-bar>
  <div class="form-container">
    <div class="noise-container">
      <mat-form-field class="noise" appearance="outline" color="primary">
        <mat-label>Noise type</mat-label>
        <mat-select [value]="noise" [(ngModel)]="noise">
          <mat-option *ngFor="let noise of noises" [value]="noise">
            {{ noise }}
          </mat-option>
        </mat-select>
      </mat-form-field>
    </div>
    <app-image (outputImageUrl)="showSelectedImage($event)" (outputSelectedImage)="selectedImage($event)"></app-image>
    <div class="buttons">
      <button *ngIf="selectedImageUrl" class="img-button" mat-raised-button color="accent" (click)="upload()">{{noise == 'None' ? 'Recognize image' : 'Add noise'}}</button>
      <button *ngIf="imageURL" class="img-button" mat-raised-button color="primary" style="margin-left: 20px" mat-stroked-button (click)="download()">Download</button>
    </div>
  </div>

  <div class="result-image-container">
    <img class="result-image" [src]="selectedImageUrl">
```

```
    <img class="result-image" style="margin-top: 10px"
[src]="imageURL">
  </div>
</div>
```

Клас MainComponent.ts представлено в лістингу Б.5.

Лістинг Б.5 – Клас MainComponent.ts

```
import {Component, OnInit} from '@angular/core';
import {FileService} from "../../services/file.service";
import {DomSanitizer, SafeUrl} from "@angular/platform-browser";
import {MessageService} from "../../services/message.service";
import {Noises} from "../../model/noises";
import {Payload} from "../../model/payload";

@Component({
  selector: 'app-main',
  templateUrl: './main.component.html',
  styleUrls: ['./main.component.scss']
})
export class MainComponent implements OnInit {

  selectedImageUrl: string | undefined;
  noise: string = Noises.NONE;

  file: File | undefined;
  loading: boolean = false;
  imageURL: SafeUrl | undefined = undefined;
  objectUrl: string | undefined = undefined;

  noises = Object.values(Noises);

  noiseName(name: string): string {
    return
Object.values(Noises)[Object.keys(Noises).indexOf(name)];
  }

  constructor(private fileService: FileService,
              private messageService: MessageService,
              private sanitizer: DomSanitizer) {
  }

  ngOnInit(): void {
  }

  upload() {
```

```
          const                                noise                        =
Object.keys(Noises)[Object.values(Noises).indexOf(this.noise         as
unknown as Noises)];
          this.recognizeImage(this.file!, noise);
        }
        recognizeImage(file: File, noise: string): void {
            this.loading = true;
            const payload: Payload = {
              file: file,
              noise: noise,
            };
            this.fileService.sendFile(payload).subscribe(
              (response) => {
                let blobFile: Blob | null = response.body;
                if (blobFile) {

                  this.objectUrl = URL.createObjectURL(blobFile);
                  this.imageURL                                             =
this.sanitizer.bypassSecurityTrustUrl(this.objectUrl);
                  this.loading = false;
                }
              },
              (error) => {
                this.loading = false;
                this.messageService.open('Failed to recognize text')
              }
            );
        }

        showSelectedImage(img: string) {
          this.selectedImageUrl = img;
          this.imageURL = undefined;
        }

        selectedImage(file: File) {
          this.file = file;
        }

        download() {
          if (this.objectUrl) {
            const anchor = document.createElement('a');
            anchor.href = this.objectUrl;
            anchor.download            =            'result.'            +
this.file!.name.split('.')[1];
            anchor.click();
            URL.revokeObjectURL(anchor.href);
          }
        }
      }
```

Файл main.component.scss представлено в лістингу Б.6.

Лістинг Б.6 – Файл main.component.scss

```
@import "~src/_variables.scss";
.main-container {
  height: 100%;
  padding: 0;
  margin: 0;
  display: flex;
  align-items: center;
  justify-content: center;
  background-color: $primary;

  .progress-bar {
    position: absolute;
    height: 10px;
    left: 0;
    top: 0;
  }

  .form-container {
    background-color: white;
    border-radius: 20px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    padding: 25px;
    margin: auto;

    .spinner {
      display: flex;
      justify-content: center;
      align-content: center;
    }

    .noise-container {
      display: flex;
      flex-direction: row;
    }

    .noise {
      width: 250px;
    }

    .buttons {
      display: flex;
      flex-direction: row;
      justify-content: space-between;
      .img-button {
        font-weight: 700;
```

```
        font-size: 20px;
      }
    }
  }

  .result-image-container {
    display: flex;
    flex-direction: column;
    padding: 10px;
    .result-image {
      max-width: 700px;
      max-height: 350px;
      object-fit: contain;
    }
  }
}
```

Клас DndDirective.ts наведено в лістингу Б.7.

Лістинг Б.7 – Клас DndDirective.ts

```
import {
  Directive,
  Output,
  Input,
  EventEmitter,
  HostBinding,
  HostListener
} from '@angular/core';

@Directive({
  selector: '[appDnd]'
})
export class DndDirective {

  @HostBinding('class.fileover') fileOver = false;
  @Output() fileDropped = new EventEmitter<File>();

  // Dragover listener
  @HostListener('dragover', ['$event'])
  onDragOver(evt: Event) {
    evt.preventDefault();
    evt.stopPropagation();
    this.fileOver = true;
  }

  // Dragleave listener
  @HostListener('dragleave', ['$event'])
  onDragLeave(evt: Event) {
```

```
    evt.preventDefault();
    evt.stopPropagation();
    this.fileOver = false;
  }

  // Drop listener
  @HostListener('drop', ['$event'])
  ondrop(evt: DragEvent) {
    evt.preventDefault();
    evt.stopPropagation();
    this.fileOver = false;
    let files = evt.dataTransfer!.files;
    if (files.length > 0) {
      this.fileDropped.emit(files[0]);
    }
  }
}
```

Файл noises.ts представлено лістингом Б.8.

Лістинг Б.8 – Файл noises.ts

```
export enum Noises {
  NONE = 'None',
  GAUSSIAN = 'Gaussian',
  SALT_PAPER = 'Salt & Paper',
  POISSON = 'Poisson',
  SPECKLE = 'Speckle'
}
```

Файл payload.ts представлено лістингом Б.9.

Лістинг Б.9 – Файл payload.ts

```
export interface Payload {
  file: File;
  noise: string;
}
```

Клас FileService.ts представлено в лістингу Б.10.

Лістинг Б.10 – Клас FileService.ts

```
import { Injectable } from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {catchError, throwError} from "rxjs";
import {Payload} from "../model/payload";

export interface ErrorModel {
  status?: number,
  message?: string
}

@Injectable({
  providedIn: 'root'
})
export class FileService {
  readonly baseUrl = 'http://localhost:5000/';

  constructor(private httpClient: HttpClient) { }

  sendFile(payload: Payload) {
    const form = new FormData();
    form.append('file', payload.file);
    form.append('noise', payload.noise);
    return this.httpClient.post(this.baseUrl, form, {responseType: 'blob', observe: 'response'})
      .pipe(catchError(this.handleError.bind(this)));
  }

  private handleError(error: { error: ErrorModel; status: any; message: any; }) {
    let result = error.error;

    if (error.error instanceof ErrorEvent) {
    } else {
      if (!error.error) {
        result = error;
      }
    }
    return throwError(() => result);
  }
}
```

Клас MessageService.ts представлено в лістингу Б.11.

Лістинг Б.11 – Клас MessageService.ts

```
import { Injectable, NgZone } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
```

```
@Injectable({
  providedIn: 'root',
})
export class MessageService {

  constructor(
    private snackBar: MatSnackBar,
    private zone: NgZone,
  ) { }

  public open(message: string, duration: number | undefined = 4000, action = 'OK'): void {
    this.zone.run(() => {
      this.snackBar.open(message, action, { duration });
    });
  }
}
```

Б.2 Лістинг серверної частини програми

Файл enums.py представлено лістингом Б.12.

Лістинг Б.12 – Файл main.py

```
from enum import Enum

class Noises(Enum):
    NONE = 'None'
    GAUSSIAN = 'Gaussian'
    SALT_PAPER = 'Salt & Paper'
    POISSON = 'Poisson'
    SPECKLE = 'Speckle'
```

Файл main.py представлено лістингом Б.13.

Лістинг Б.13 – Файл main.py

```
import re
import cv2
import numpy as np
import imutils
import easyocr
import pytesseract
```

```
import random
import skimage
from flask import Flask, request, make_response, abort
from flask_cors import CORS
from model.enums import Noises
app = Flask(__name__)
CORS(app)

def gaussian(img):
    noisy_img = skimage.util.random_noise(img, mode="gaussian")
    return noisy_img

def salt_paper(img):
    # Отримання розмірів зображення
    h, w, c = img.shape

    # Випадковий вибір декількох пікселів у
    # зображенні, щоб пофарбувати їх у білий колір
    # Вибір випадкового число від 1000 до 10000
    number_of_pixels = random.randint(1000, 10000)
    for i in range(number_of_pixels):
        # Вибір випадкової координати Y
        y_coord = random.randint(0, h - 1)
        # Вибір випадкової координати X
        x_coord = random.randint(0, w - 1)
        # Зафарбовування цього пікселя у білий колір
        img[y_coord][x_coord] = 255
    # Випадковий вибір декількох пікселів у
    # зображенні, щоб пофарбувати їх у чорний колір
    # Вибір випадкового число від 300 до 10000
    number_of_pixels = random.randint(300, 10000)
    for i in range(number_of_pixels):
        # Вибір випадкової координати Y
        y_coord = random.randint(0, h - 1)
        # Вибір випадкової координати X
        x_coord = random.randint(0, w - 1)
        # Зафарбовування цього пікселя у чорний колір
        img[y_coord][x_coord] = 0
    return img

def poisson(img):
    noisy_img = skimage.util.random_noise(img, mode="poisson")
    return noisy_img

def specle(img):
    mean = 0
    var = 0.25
    sigma = np.sqrt(var)
    gauss = np.random.normal(mean, sigma, img.size)
    gauss     =     gauss.reshape(img.shape[0],     img.shape[1],
img.shape[2]).astype('uint8')
    noise = img + img * gauss
    return noise
```

```
def recognize(img):
    # Нормалізація зображення
    img = cv2.normalize(src=img, dst=None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8U)
    # Переведення до сірого формату
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    # Накладання фільтру
    filtered = cv2.medianBlur(gray, 3)
    imgFilter = filtered
    # Знаходження країв зображення
edges = cv2.Canny(imgFilter, 50, 200)
    # Знаходження контурів зображення
    cont = cv2.findContours(edges.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    # Зчитування контурів
    cont = imutils.grab_contours(cont)
    # Сортування контурів
    cont = sorted(cont, key=cv2.contourArea, reverse=True) # сорриpовка контуров
    pos = None
    for c in cont:
        # Знаходження закритих контурів
        approx = cv2.approxPolyDP(c, 8, True)
        # Якщо довжина складається з 4 елементів - координати 4 кутів квадрата
        if (len(approx)) == 4:
            pos = approx
            break
# Формування розміру квадрату для обведення номерного знаку
    if pos[0][0][0] < pos[0][0][1] and pos[1][0][0] < pos[1][0][1] \
            and pos[2][0][0] < pos[2][0][1] \
            and pos[3][0][0] < pos[3][0][1]:
        pos[0][0][0] = pos[0][0][0] - 70
        pos[1][0][0] = pos[1][0][0] - 70
        pos[2][0][0] = pos[2][0][0] + 70
        pos[3][0][0] = pos[3][0][0] + 70
    elif pos[0][0][0] > pos[0][0][1] and \
            pos[1][0][0] > pos[1][0][1] \
            and pos[2][0][0] > pos[2][0][1] \
            and pos[3][0][0] > pos[3][0][1] \
            and (pos[3][0][0] - pos[3][0][1]) < 150 \
            and (pos[1][0][0] - pos[1][0][1]) < 150:
        pos[0][0][0] = pos[0][0][0] - 70
        pos[1][0][0] = pos[1][0][0] - 70
        pos[2][0][0] = pos[2][0][0] + 70
        pos[3][0][0] = pos[3][0][0] + 70
    elif pos[0][0][0] > pos[0][0][1] and \
            pos[1][0][0] > pos[1][0][1] \
            and pos[2][0][0] > pos[2][0][1] \
            and pos[3][0][0] > pos[3][0][1] \
            and (pos[3][0][0] - pos[3][0][1]) > 150 \
```

```
                and (pos[1][0][0] - pos[1][0][1]) > 150:
            pos[0][0][0] = pos[0][0][0] + 70
            pos[1][0][0] = pos[1][0][0] - 70
            pos[2][0][0] = pos[2][0][0] - 70
            pos[3][0][0] = pos[3][0][0] + 70

        elif   pos[0][0][0]   <   pos[0][0][1]   and   pos[3][0][0]   <
pos[3][0][1]:
            pos[0][0][0] = pos[0][0][0] - 70
            pos[1][0][0] = pos[1][0][0] + 70
            pos[2][0][0] = pos[2][0][0] + 70
            pos[3][0][0] = pos[3][0][0] - 70
        elif   pos[0][0][0]   >   pos[0][0][1]   and   pos[3][0][0]   >
pos[3][0][1]:
            pos[0][0][0] = pos[0][0][0] + 70
            pos[1][0][0] = pos[1][0][0] - 70
            pos[2][0][0] = pos[2][0][0] - 70
            pos[3][0][0] = pos[3][0][0] + 70
        # Виділеного знайденого контуру
        mask = np.zeros(gray.shape, np.uint8)
        # Рисування контуру
        cv2.drawContours(mask, [pos], 0, 255, -1)
        # виділення контуру знайденого номерного знаку
        # з початкового зображення
        bitwise_img = cv2.bitwise_and(img, img, mask=mask)
        # Вирізання знайденого номерного знаку
        (x, y) = np.where(mask == 255)
        (x1, y1) = (np.min(x), np.min(y))
        (x2, y2) = (np.max(x), np.max(y))
        gr = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        cropp = gr[x1:x2, y1 - 30:y2 + 70]
        # Накладання фільтру
        imgFilter = cv2.bilateralFilter(cropp, 11, 30, 50)
        # Збільшення номерного знаку
        cropp    =    cv2.resize(imgFilter,    None,    fx=3,    fy=3,
interpolation=cv2.INTER_AREA)
        # Накладання фільтру
        blur = cv2.GaussianBlur(cropp, (7, 7), 0)
        # Збільшення чіткості зображення
        kernel = np.array([[-1, -1, -1], [-1, 9, -1], [-1, -1, -1]])
        im = cv2.filter2D(blur, -1, kernel)
    # Зчитування тексту з номерного знаку
        text = easyocr.Reader(['en'])
        text = text.readtext(im)
        print(text)
        if len(text) == 0:
            abort(400)
        # Деякі коди номерних знаків євпропейських країн
        europeLicencePlates   =   {"Austria":   "A",   "Belgium":   "B",
"Germany": "D", "Latvia": "LV", "Lithuania": "LT",
                                "Romania":   "RO",   "Poland":   "PL",
"Finland": "FIN", "Hungary": "H"}
```

```
        # Корекція деяких помилок в кодах країн при розпізнаванні
        countryOnPlate = text[0][-2].upper()
        print("countryOnPlate", countryOnPlate)
        if countryOnPlate.upper() == "LI" or countryOnPlate.upper()
== "T":
            countryOnPlate = "LT"
        elif countryOnPlate == "8":
            countryOnPlate = "B"
        elif countryOnPlate == "0":
            countryOnPlate = "B"
        elif countryOnPlate == "FL":
            countryOnPlate = "PL"
        elif countryOnPlate == "FINI":
            countryOnPlate = "FIN"

        resultText = []
        for index, i in enumerate(text):
            for j in [1]:
                resultText.append(i[j].upper())
        country = ""
        for item in europeLicencePlates.items():
            if countryOnPlate == item[1]:
                country = item[0]
        if country != "":
            resultText.pop(0)
        str1 = ""
        str = ""
        if country == "":
            str = re.sub("\W", "", str1.join(resultText), 0,
re.IGNORECASE)
        else:
            str = country + ", " + re.sub("\W", "",
str1.join(resultText), 0, re.IGNORECASE)
        # Формування розпізнаного тексту на зображенні
        final_img = cv2.putText(img, str, (y1, x2 + 30),
cv2.FONT_HERSHEY_PLAIN, 2, (245, 221, 66), 2)
        # Обведення контуру номерного знаку
    final_img = cv2.rectangle(img, (y1, x1), (y2, x2), (0, 255, 0),
2)
        return final_img

    @app.route('/', methods=['POST'])
    def index():
        r = request
        noise = Noises[r.form.get('noise')]
        file = r.files['file']
        img_str = file.read()
        nparr = np.fromstring(img_str, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        extension = file.filename.split(".")[1].lower()
        result_image = None
```

```
        if noise == Noises.GAUSSIAN:
            result_image = gaussian(img)
            result_image   =   cv2.normalize(result_image,   dst=None,
alpha=0, beta=255, norm_type=cv2.NORM_MINMAX,
                                          dtype=cv2.CV_8U)
        elif noise == Noises.SALT_PAPER:
            result_image = salt_paper(img)
        elif noise == Noises.POISSON:
            result_image = poisson(img)
            result_image   =   cv2.normalize(result_image,   dst=None,
alpha=0, beta=255, norm_type=cv2.NORM_MINMAX,
                                          dtype=cv2.CV_8U)
        elif noise == Noises.SPECKLE:
                result_image = specle(img)
        else:
            result_image = recognize(img)
    is_success,   im_buf_arr   =   cv2.imencode("."   +   extension,
result_image)
        byte_im = im_buf_arr.tobytes()

        response = make_response(byte_im)
        response.headers.set('Content-Type', 'image/' + extension)
        response.headers.set('Content-Disposition',      'attachment',
filename="result" + "." + extension)
        return response

    if __name__ == "__main__":
        app.run()
```