КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

СИСТЕМА АВТОМАТИЗОВАНОЇ ОБРОБКИ ФІНАНСОВИХ ДОКУМЕНТІВ

Дипломна робота магістра
студента 2 року навчання
спеціальність: 123 «Комп'ютерна інженерія»
Максима САЛО

Науковий керівник асистент кафедри
комп'ютерної інженерії
Юрій ЮРЧИК

Рецензент **Лебедев Денис Юрійович** к.т.н., доцент кафедри конструювання електронно-обчислювальної апаратури, факультет електроніки, Київський політехнічний інститут імені Ігоря Сікорського

До захисту допускаю

Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри "____" _____ 2022 р., протокол № ___

Київ - 2022

РЕФЕРАТ

Обсяг роботи за об’ємом складає 41 сторінку, містить 4 рисунки, 2 таблиці, 6 додатків, використано 12 інформаційних джерел.

Ключові слова: OCR, зображення, документ, сервіс

Предмет дослідження: спосіб автоматичної обробки фінансових документів за допомогою алгоритмів оптичного розпізнавання символів та їх верифікація.

Актуальність обраної теми полягає в тому, що потік паперових фінансових документів у організаціях та компаніях дуже великий. І саме ця умова змушує розробити систему, яка автоматизує частину процесів. Це необхідно для того, щоб скоротити час, необхідний для обробки документів із введенням їх у електронну систему обліку.

Метою роботи є дослідження способів ефективного розпізнавання даних з фото або сканованих зображень фінансових документів на прикладі квитанцій про оплату та перевірка їх справжності.

Результати роботи : проведено порівняльний аналіз алгоритмів та програмних бібліотек для оптичного розпізнавання даних фінансових документів, досліджено способи покращення результатів розпізнавання, розроблено програмне забезпечення для розпізнавання даних та перевірки справжності фінансових документів на прикладі банківських квитанцій про оплату.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ 5

ВСТУП 6

РОЗДІЛ 1. Принцип роботи системи автоматизованої обробки фінансових документів 8

1.1 Вимоги до системи автоматизованої обробки фінансових документів 8

1.2 Обмін даними між компонентами САОФД 10

1.3 ПЗ Клієнтської частини САОФД 11

1.4 ПЗ Серверної частини САОФД 11

1.5 ПЗ автоматичної обробки ФЗ або СЗ 12

1.6 ПЗ для обміну даними з системою обліку 12

РОЗДІЛ 2. Способи розпізнавання даних фінансових документів 13

2.1 Оптичне розпізнавання символів 13

2.2 Шаблонний метод 14

2.3 Метод з використанням ознак 14

2.4 Структурний метод 15

2.5 Штучна нейронна мережа 15

2.6 Проблеми оптичного розпізнавання текстів 15

2.7 Процес розпізнавання даних з зображень фінансових документів 16

2.8 Попередня обробка зображення ФД 16

2.9 Виокремлення корисної інформацію з тексту ФД 18
РОЗДІЛ 3. Розробка системи автоматичної обробки фінансових документів 21
3.1 Компоненти системи 21
3.2 Інструменти реалізації компонентів САОФД 22
3.3 Засоби розпізнавання тексту 24
3.4 Стек технологій 25
3.5 Серверна частина 26
3.6 Сервіс обробки зображення та розпізнавання тексту 30
3.5 Алгоритм попередньої обробки зображення 30
3.6 Алгоритм виокремлення ідентифікаційного коду квитанції. 32
3.7 Перевірка результатів роботи системи 32
ВИСНОВКИ 33
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА 34
ДОДАТКИ 35

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

САОФД - Система атоматизованої обробки фінансових документів

OCR - Optical character recognition (Оптичне розпізнавання символів)

ФД - Фінансовий документ

ПЗ - Програмне забезпечення

СЗ - Скановане зображення

ФЗ - Фото зображення

ВСТУП

Оперативна та якісна обробка фінансових документів, таких як документи про оплату – важливе питання для кожного підприємства.

Але на даному етапі розвитку технологій в нашій країні повністю відмовитися від використання паперових документів неможливо, все ще більшість компаній працюють з паперовими документами.

В даний час багато компаній прагнуть перевести документообіг в електронний вигляд. Рішення з використанням систем автоматизованої обробки фінансових документів є особливо актуальним для компаній та організацій які часто виконують фінансові операції або надають платні послуги, оплата яких підтверджується документом про оплату.

Відомо, що рахунки на оплату є одним із найпоширеніших типів документації для кожної компанії. Навіть підприємства середнього розміру обробляють тисячі платіжних документів щомісяця. З огляду на нинішній рівень розвитку технологій автоматизованої обробки документів, їх ручне опрацювання є економічно невигідним, трудомістким та загалом неефективним заняттям.

Автоматизація бухгалтерського обліку можлива завдяки створенню системи автоматизованої обробки фінансової документації.

Автоматизація вводу даних проводиться для фінансової документації, яка засвідчує проведення банківської операції. За підтримки плагінів (модулів розширення) можна розширювати список утримуваних документів.

Процес від сканування до збереження даних в облікову систему, в порівнянні з ручною обробкою документів, яка може займати кілька днів, займає всього кілька хвилин. Оскільки обробка пакетів документів проходить

паралельно, діяльність робітників компанії скорочується. Завдяки автоматизації трудові ресурси націлюються на врегулювання і вирішення більш актуальних і важливих завдань, аніж обробку і внесення даних вручну з клавіатури робітником до облікової системи.

РОЗДІЛ 1. Принцип роботи системи автоматизованої обробки фінансових документів

1.1 Вимоги до системи автоматизованої обробки фінансових документів

Автоматизована обробка фінансових документів – це процес автоматичного зчитування даних документу за допомогою програмного забезпечення автоматизованої обробки без прямої участі людини із збереженням розпізнаних даних до системи обліку.[1]

Система автоматизованої обробки фінансових документів повинна являти собою ПЗ, яке інтегрується в електронні системи керування та обліку організацій чи компаній. Шляхом фотографії або потокового сканування паперові документи оцифровуються, потім відправляються до САОФД. Системою обробки розпізнаються дані з ФЗ або СЗ, ці дані вносяться до облікової системи, а копії ФЗ або СЗ розміщуються в електронному сховищі. Весь процес відбувається автоматично за мінімальної участі людини.

Для легкої інтеграції в уже існуючі системи обліку ПЗ автоматичної обробки фінансових документів повинно бути представлене у вигляді прикладного програмного інтерфейсу API, запущеного в програмному середовищі уже існуючої системи обліку або у вигляді ізольованого програмного середовища (контейнера). Для передачі оброблених даних системі обілку ПЗ САОФД повинно мати доступ до ресурсів системи обліку (баз даних, API тощо).

Загалом схема системи автоматичної обробки фінансових документів виглядає наступним чином (Рис.1) :

- ПЗ (клієнт) для відправки ФЗ або СЗ

- ПЗ (сервер) яке отримує ФЗ або СЗ від клієнта та створює задачу автоматичної обробки
- ПЗ автоматичної обробки ФЗ або СЗ
- ПЗ для обміну даними з системою обліку

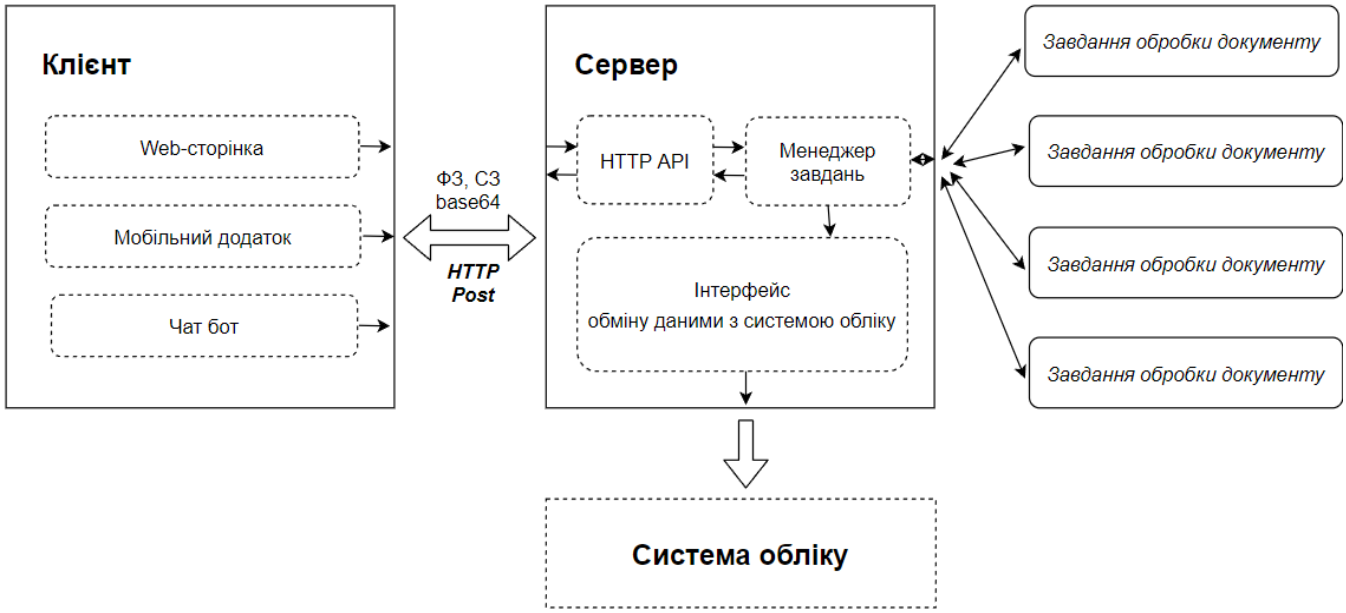


Рис. 1. Схема системи автоматичної обробки фінансових документів.

Загалом процес обробки документів включає в себе наступні кроки:

1. До системи за допомогою користувацького інтерфейсу надсилається скановане або фото зображення документу.
2. Зображення документу надсилаються на сервер розпізнавання, де відбувається автоматичне розпізнавання вмісту документа.
3. Після успішного розпізнавання вмісту, отримані данні збираються в єдину структуру та відправляються до облікової системи компанії.

1.2 Обмін даними між компонентами САОФД

Обмін даними між окремими складовими системи (Клієнт – Сервер – Система обліку) відбувається за допомогою використання прикладних програмних інтерфейсів API та підходу REST.

Прикладний програмний інтерфейс[2] або API — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Це набір чітко визначених методів для взаємодії різних програмних компонентів чи середовищ. API надає засоби для швидкої розробки програмного забезпечення. API може бути використаний для web-застосувань, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

REST (англ. Representational State Transfer)[3] — це архітектурний стиль для розподілених систем, які надають доступ до інформаційних ресурсів. Стиль REST вперше був описаний і популяризований у 2000 році Роєм Філдінгом, який є одним з творців протоколу HTTP. В основі REST закладено web-мережі і, зокрема, можливості HTTP. За правилами архітектури REST дані повинні передаватися у вигляді невеликої кількості стандартних форматів, наприклад, XML або JSON. HTTP REST повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь». Такий підхід забезпечує масштабовність системи і дозволяє її змінювати відповідно до нових вимог.

HTTP[4] — протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від HyperText Transfer Protocol, протокол передачі гіпертекстових документів.

1.3 ПЗ Клієнтської частини САОФД

ПЗ клієнтської частини являє собою web-сторінку, мобільний додаток або чат-бот, який надає можливість зробити фото або скановане зображення документа та відправити його на сервер автоматичної обробки документів. Клієнт відображає повідомлення про статус процесу обробки та помилки, якщо вони відбудуться при виконанні процесу обробки.

1.4 ПЗ Серверної частини САОФД

ПЗ серверної частини являє собою процес, що виконується в програмному середовищі системи обліку або в ізольованому середовищі. Сервер має прикладний програмний інтерфейс API для отримання ФЗ або СЗ від клієнта за допомогою HTTP запитів. При отриманні ФЗ або СЗ сервер створює окреме завдання (ізольований процес), який виконує алгоритм розпізнавання корисних даних з ФЗ або СЗ. Ізольований процес використовується для можливості горизонтального масштабування та зменшення навантаження на сервер. Оскільки алгоритм розпізнавання є складним з точки зору навантаження на обчислювальні ресурси, то необхідно надати можливість створювати процеси з алгоритмом розпізнавання на нових обчислювальних ресурсах, якщо поточних обчислювальних ресурсів недостатньо, тобто використати підхід розподілу навантаження. Сервер може керувати процесами розпізнавання. Якщо обчислювальних ресурсів не вистачає для створення нового процесу розпізнавання, то останній ставиться в чергу і сервер про це інформує клієнтську частину. При запиті статусу виконання обробки ФЗ чи СЗ клієнтом, сервер запитує про статус процесу обробки і відправляє цей статус клієнтові.

1.5 ПЗ автоматичної обробки ФЗ або СЗ

ПЗ автоматичної обробки ФЗ або СЗ являє собою процес атомарної операції обробки документу, який на вході отримує зображення, виконує алгоритм розпізнавання тексту і виокремлення корисних даних та повертає ці дані на виході. Процес має прикладний програмний інтерфейс, що дає змогу серверу отримувати дані про статус виконання процесу.

1.6 ПЗ для обміну даними з системою обліку

ПЗ для обміну даними з системою обліку являє собою набір функцій, який дає змогу інтегрувати ПЗ обробки фінансових документів в уже існуючі системи обліку. Обмін даними з системою обліку може бути реалізовано декількома способами:

- За допомогою API системи обробки: система має набір HTTP методів, які дозволяють запитувати оброблені дані, статус їх обробки, копії квитанцій тощо.
- За допомогою використання зворотних викликів webhook: система обробки в процесі за допомогою HTTP за налаштованим посиланням надсилає сповіщення та оброблені дані до системи обліку.
- За допомогою прямого доступу до бази даних системи обліку: система обробки, використовуючи налаштований модуль роботи з базою даних системи обліку, записує дані безпосередню в БД в визначеному системою обліку форматі.

РОЗДІЛ 2. Способи розпізнавання даних фінансових документів

Основною задачею системи є розпізнавання та виокремлення даних з зображення. Тому постає питання, яким чином можливо отримати текст з зображення та на основі цього тексту отримати корисні дані?

2.1 Оптичне розпізнавання символів

Розпізнавання тексту на зображеннях або оптичне розпізнавання символів[5] (англ. optical character recognition, OCR) - один з напрямків розпізнавання образів, завдання якого полягає в перекладі зображень рукописного, машинного або друкованого тексту в текстові дані, що використовуються для подальшої обробки комп’ютерними системами.

Розпізнавання тексту на зображеннях є важливим завданням машинного навчання, оскільки це дозволяє організувати зручну взаємодію з даними: редагування, аналіз, пошук слів чи фраз тощо.

Проблему оптичного розпізнавання символів вирішують кілька останніх десятиліть. Завдяки швидкому розвитку комп’ютерних технологій було розроблено ефективні методи оптичного розпізнавання, які використовуються в алгоритмах автоматизованої обробки документів.

Алгоритми оптично розпізнавання символів використовують основні методи[6]:

- Шаблонний метод
- Метод з використанням ознак
- Структурний метод
- Штучна нейронна мережа

2.2 Шаблонний метод

Одним з перших методів розпізнавання символів був шаблонний метод. Основою цього методу є порівняння вхідного зображення символу і деякої кількості шаблонних зображень з символами, що розпізнаються. При цьому розмір вхідного і шаблонного зображення має бути однаковим. Для цього виконують нормалізацію вхідного зображення.

Для ідентифікації символу розраховуються спеціальні коефіцієнти кореляції, які показують ступінь відмінності вхідного зображення і шаблонів. При цьому, якщо один з коефіцієнтів кореляції набагато більший за інші, то символ на вхідному зображенні точно ідентифікується.

Перевагою даного методу є простота реалізації алгоритму, відносно висока швидкість розпізнавання, а також ефективне розпізнавання символів з різними дефектами.

Недоліком даного методу є необхідність мати набір шаблонів символів того ж шрифту, що і розпізнається. Це робить даний метод менш універсальним та гнучким [7].

2.3 Метод з використанням ознак

В методі з використанням ознак виконується побудова n мірного вектора ознак, наприклад, кількість замкнутих областей, відсоток заповнення і т.д. Побудований вектор ознак порівнюється з набором еталонних векторів тієї ж розмірності.

Перевагою даного метода є ефективне розпізнавання символів незалежно від їх нахилу, пропорцій тощо.

Недоліком даного методу є мала стійкість до дефектів символів [7].

2.4 Структурний метод

При структурному методі розпізнавання вхідне зображення перетворюється в топологічне представлення, яке описує взаємне розташування структурних елементів символу. Найчастіше ця інформаціє представляється у вигляді графу.

Перевагою даного метода є мала чутливість до розмірів, нахилу, пропорцій символів.

Недоліком даного методу є мала стійкість до дефектів символів [7].

2.5 Штучна нейронна мережа

Одним за найбільш поширених методів розпізнавання символів є розпізнавання за допомогою штучної нейронної мережі. Основна відмінність використання нейронної мережі полягає в тому, що нейронні мережі не програмуються на розпізнавання символів, а навчаються.

Перевагою даного методу є висока ефективність і швидкодія.

Недоліком методу є необхідність у великій кількості даних, які використовуються для навчання нейронної мережі і складність розробки [7].

2.6 Проблеми оптичного розпізнавання текстів

Алгоритми оптичного розпізнавання текстів мають ряд факторів, які негативно впливають на вихідний результат.

Розглянемо фактори зображення, які негативно впливають на ефективність розпізнавання:

- Різноманітність форм зображення: документ може містити кілька шрифтів відразу, а символи можуть бути схожі за зображенням.

- Спотворене зображення, що містить текст, наявність шумів, погана якість вхідного зображення.
- Варіації розмірів, масштабу та положення символів на сторінці.
- Вплив вихідного масштабу друку: система оптичного розпізнавання тексту повинна бути нечутливою по відношенню до способу верстки, відстані між рядками та іншими параметрами друку.

Тому для підвищення ефективності розпізнавання використовують попередню обробку зображення та комбінацію методів розпізнавання.

Комбінація методів розпізнавання або «адаптивний метод» дозволяє використовувати комбінації методів розпізнавання символів та підстроювати алгоритм розпізнавання під різні характеристики зображення вхідного тексту.

2.7 Процес розпізнавання даних з зображень фінансових документів

Процес розпізнавання корисних даних з зображень фінансових документів включає в себе наступні етапи:

- Попередня обробка зображення ФД.
- Розпізнавання тексту.
- Виокремлення корисної інформацію з тексту.

2.8 Попередня обробка зображення ФД

Попередня обробка зображення – це алгоритм, який включає в себе ряд операцій над вхідним зображенням, для максимального наближення цього зображення до «ідеального» для розпізнавання зображення. Вона використовується для збільшення ефективності алгоритму розпізнавання, коли на вхід подається «неякісне» зображення. Попередня обробка зображень підвищує ефективність розпізнавання.

Попередня обробка включає в себе ряд послідовних перетворень вхідного зображення[8]:

- Отримання зображення в сірих кольорах.
- Розмиття зображення та фільтрація шумів
- Отримання границь елементів зображення
- Пошук найбільшого контуру документу на зображенні
- Вирівнювання перспективи
- Обрізання непотрібного фону
- Отримання бінарного зображення

Отримання зображення в сірих кольорах зменшує розмірність вхідних векторів зображення, що дає змогу ефективніше обробляти зображення в подальшому.

Розмиття зображення разом з алгоритмом фільтрації дозволяє істотно зменшити кількість шумів на зображенні. Шуми негативно впливають на подальшу обробку зображення (пошук границь і контурів) і розпізнавання тексту, тому їх видалення є необхідним.

Отримання границь зображення дозволяє знайти всі границі об’єктів на зображенні, які можуть бути контуром самого документу. Дана операція застосовується коли на зображенні документу присутній додатковий фон. Для досягнення максимального результату даної операції фон повинен бути однорідним.

Після отримання границь об’єктів відбувається пошук найбільшого за периметром чотирикутного контуру. Цей контур є границею документа на зображенні.

Коли контур знайдено, за його вершинами відбувається вирівнювання перспективи, тобто чотирикутний контур перетворюється на прямокутний і відповідно вирівнюється перспектива зображення. Далі за межами цього контуру відсікається непотрібний фон.

Останнім кроком попередньої обробки є отримання бінарного зображення, тобто зображення, яке має тільки два кольори: білий і чорний.

Після попередньої обробки ми отримуємо зображення документа, яке дещо нагадує зображення створене сканером.

2.9 Виокремлення корисної інформацію з тексту ФД

Алгоритм оптичного розпізнавання тексту повертає дані у вигляді тексту або у вигляді структур даних, які містять в собі структурований за блоками текст. Виділити корисні дані з розпізнаного тексту можна за допомогою декількох методів[9]:

- використання регулярних виразів
- аналіз контексту окремих блоків
- використання нейронних мереж
- комбінування методів

Метод з використанням регулярних виразів найпростіший в реалізації та застосовується якщо потрібно виокремити дані, що мають однакову структуру і формат в різних ФД. Наприклад, якщо фінансовим документом, що обробляється є квитанція про оплату і корисними даними є ідентифікатор квитанції, то можна використати регулярний вираз. Розглянемо випадок, коли відомо, що ідентифікаційний код квитанціє про оплату – це рядок, який складається з чотирьох груп по 4 символи (букви та цифри), які можуть бути

розділені дефісом або не мати розділювального символу (Рис. 2). В такому випадку нам підійде регулярний вираз наступного вигляду (Рис. 2):

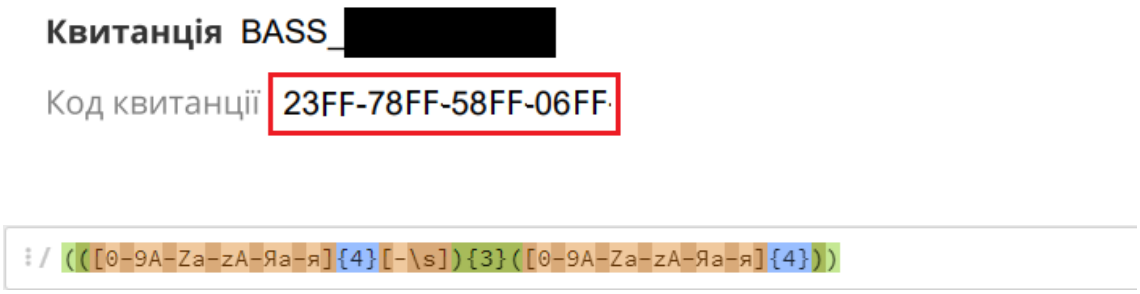


Рис.2 Приклад ідентифікаційного коду квитанції та регулярного виразу для його розпізнавання

За даним регулярним виразом в тексті буде знайдено ідентифікаційний код квитанції, який є корисними даними.

Аналіз контексту окремих блоків полягає в обробці окремих блоків тексту, знаходження блоків, які містять ключові слова, знаходження рядків, які містять ключові слова та за знайденими ключовими словами виділення корисної інформації (Рис. 3).

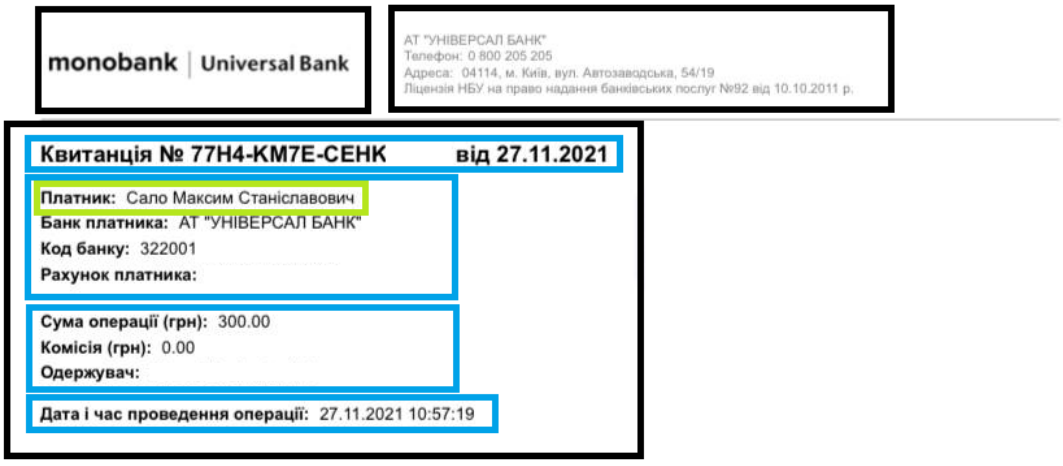


Рис.3 Приклад аналізу тексту зображення та знаходження даних платника за блоками

Для аналізу контексту блоків потрібно мати набір ключових слів. Розглянемо приклад виділення інформації про ПІБ платника. Алгоритм розпізнавання тексту може повернути дані у вигляді суцільного тексту або у вигляді блоків. Якщо отримано суцільний текст, то проводиться пошук за ключовими словами у кожному рядку і виділяється n кількість слів після знайденого ключового слова і символу початку ПІБ. У випадку з знаходженням ПІБ платника ключовим словом є рядок «платник» і наступні 3 слова власне і є текстом, який містить ПІБ платника. В якості символу розділення слів обрано символ пробілу.

У випадку коли отримано текст у вигляді блоків, рекурсивно у кожному блоці відбувається пошук необхідного рядка за ключовими словами і як в попередньому випадку з рядка виділяється необхідний текст, що містить дані про ПІБ платника.

При методі з використанням нейронних мереж маємо декілька нейронних мереж, які навчені на наборі фінансових документів різних призначень та організацій. Кожна з нейронних мереж відповідає за пошук цілісної одиниці корисної інформації, наприклад, ПІБ, дата, ідентифікаційний код, назва організації тощо. Коли нейронні мережі повернули дані, вони компонуються в структурований об’єкт даних. Даний метод є найскладнішим у реалізації, адже, окрім розробки нейронних мереж, вимагає навчання нейронних мереж на підготовлених наборах даних ФД.

Якщо під час виконання алгоритмів виокремлення даних не знайдено, користувач інформується про це і має перевірити документ на наявність необхідних даних або повторити спробу з ФЗ або СЗ документу кращої якості.

РОЗДІЛ 3. Розробка системи автоматичної обробки фінансових документів

Під час виконання кваліфікаційної роботи магістра було розроблено систему автоматизованої обробки фінансових документів, яка надає змогу користувачам після оплати послуг компанії надіслати в систему фото квитанції про оплату для її підтвердження. Основною задачею даної системи є розпізнавання ідентифікаційного коду квитанції для його подальшої перевірки через Державний сервіс перевірки квитанцій.

3.1 Компоненти системи

Розроблена система автоматизованої обробки фінансових документів складається з компонентів:

- Клієнтська частина - web-сторінка та мобільний додаток
- Серверна частина – Node.js сервер
- Сервіс обробки зображення та розпізнавання тексту

За допомогою клієнтської частини користувачі мають змогу зробити фото квитанції, завантажити ФЗ або СЗ квитанції та відправити його на сервер компанії для подальшої обробки.

Серверна частина відповідає за отримання документів, їх збереження, створення завдань обробки, обмін даними з системою обліку. Серверна частина має відкритий HTTP API інтерфейс, який дозволяє клієнтові відправляти ФЗ або СЗ в форматі base64, а також отримувати інформацію про статус обробки документу. На серверній частині реалізовано сервіс, який дозволяє запускати процеси попередньої обробки зображення та розпізнавання тексту. Кожен такий процес запускається у вигляді окремого потоку, що дозволяє зменшити навантаження на обчислювальні ресурси системи.

3.2 Інструменти реалізації компонентів САОФД

Для реалізації серверної частини системи автоматизованої обробки фінансових документів було обрано мову Javascript та інструменти реалізації серверних застосувань Node.js Express.

Node.js робить JavaScript мовою для вирішення різного роду завдань, а не тільки для створення веб-сторінок. Також Node.js називають середовищем виконання мови Javascript. Node.js зв'язує додаткові бібліотеки, налаштовує процеси зборки застосувань, виконує роль HTTP-сервера, може викликати функції написані іншими мовами. Простіше кажучи, цей інструмент додає функціонал, який дозволяє створювати повноцінне ПЗ або системи програмних застосувань чи сервісів [10]. Node.js середовище може бути запущено в ізольованому середовищі (контейнері), що дає змогу за допомогою декількох команд інтегрувати ПЗ в існуючі системи, а також розподіляти навантаження при виконанні складних з точки зору використання ресурсів процесів.

За допомогою Node.js Javascript переводиться у машинний код — набір інструкцій, які виконуються комп’ютером без інтерпретації, тому код стає швидким.

Оскільки клієнтська частина може бути представленою у вигляді мобільного додатку або web-сторінки, постає проблема у використанні мультиплатформових інструментів реалізації застосувань, які би дали змогу використовуючи одну кодову базу створювати і мобільні додатки, і web-застосування. Також необхідною є можливість роботи з зображеннями. На сьогодні існує кілька інструментів з відкритим кодом, які дозволяють створювати мультиплатформові застосування, зокрема Flutter, React, Cordova (Таблиця 1).

Таблиця 1. Порівняння деяких інструментів розробки клієнтських застосувань.

Інструмент	**Flutter**	**React**	**Cordova**
Підтримувані платформи	Android, IOS, Windows, Linux, Mac, Web.	Android, IOS, Web	Android, IOS
Типи застосувань	Мобільні та десктопні додатки, веб-сайти	Мобільні додатки, веб-сайти	Мобільні додатки
Кодова база	Єдина	Потребує деяких змін в залежності від платформи	Єдина
Мова	Dart	Javascript	Javascript

Для вирішення даної задачі було обрано використання інструментів розробки Flutter та мову Dart, оскільки це єдиний інструмент, який дозволяє одночасно розробляти ПЗ для різних платформ, у тому числі і web-застосувань, використовуючи єдину кодову базу.

Flutter — це набір інструментів розробника і фреймворк із відкритим програмним кодом для створення додатків для різних платформ (Android, iOS, Windows, Linux, Mac та ін..) а також для Web [11]. Flutter для написання коду використовує мову високого рівня dart, яка інтерпретується в інші мови низького рівня залежно від платформи на якій запускається. Flutter та Dart мають велику кількість бібліотек та розширень з відкритим кодом, які вирішують всі основні задачі реалізації клієнтських застосувань.

Для реалізації клієнтської частини у вигляді чат-боту було використано інструменти розробки ботів Telegram.

3.3 Засоби розпізнавання тексту

Сьогодні для розпізнавання тексту існує велика кількість реалізованих бібліотек. Серед них найбільш популярними є Tesseract, Gocr, Copyfish, OCR.Space, Iron (Таблиця 2).

Таблиця 2. Порівняння деяких бібліотек оптичного розпізнавання символів.

Бібліотека	**Tesseract**	**Gocr**	**OCR.Space**	**Iron**
Підтримувані платформи	Windows, Linux, Mac	Windows, Linux	Всі з підтримкою HTTP	Всі з підтримкою HTTP
Середовище виконання	Локальне	Локальне	Хмараний сервіс	Хмарний сервіс
Використання	Безкоштовне	Безкоштовне	Безкоштовне	Платне
Мови програмування	Javascript, Python, C++, .NET	C	Будь-які з можливістю роботи з HTTP	Будь-які з можливістю роботи з HTTP

Всі ці бібліотеки використовують адаптивний метод розпізнавання та мають ряд налаштувань алгоритму розпізнавання. З огляду на популярність і кількість підтримуваних мов та операційних систем Tesseract є найперспективнішою бібліотекою для OCR.

Tesseract — це бібліотека оптичного розпізнавання символів з підтримкою різних мов програмування. Дана бібліотека розповсюджується під ліцензією Apache. Tesseract спочатку був розроблений компанією Hewlett-Packard для

вирішення власних задач компанії у 1980-х роках. У 2005 було випущено бібліотеку з відкритим кодом, а з 2006 року розвиток інструментів Tesseract OCR спонсорує компанія Google [12]. Проблемою даної бібліотеки є те, що вихідні дані розпізнавання мають дуже низьку якість, якщо вхідне зображення не було попередньо оброблено. Зображення мають бути збільшені таким чином, щоб мінімальна висота символу вхідного тексту становила 20 пікселів, зображення не мало відхилення за перспективою, а також мало високий контраст.

3.4 Стек технологій

Для реалізації системи автоматизованої обробки фінансових документів було обрано наступний стек технологій:

- Node.js – мова написання основної логіки сервісу синхронізації новин (оскільки отримувані дані мають динамічну структуру, та передаються у вигляді json, то зручно буде їх опрацьовувати саме слабо типізованою мовою Javascript)
- Docker – програмне забезпечення для автоматизації розгортання і керування контейнерами.
- Tesseract – бібліотека яка надає інструменти для оптичного розпізнавання тексту (OCR).
- Express.js – фреймворк для створення веб за стосунків.
- OpenCV – бібліотека алгоритмів комп’ютерного зору та обробки зображень.
- Flutter – фреймворк для мультиплатформової розробки застосувань з користувацьким інтерфейсом.

Такий стек технологій обрано з міркувань про те, що він підтримується більшістю програмних середовищ та операційних систем, дозволяє створювати продуктивні легковагові та масштабовані системи, які дуже добре інтегруються в уже існуючі системи.

Алгоритм розпізнавання розроблено з використанням бібліотек OpenCV та Tesseract.js.

Бібліотека OpenCV використовується для попередньої обробки зображення. Дана бібліотека має в собі набір методів для обробки вхідного зображення.

Бібліотека Tessseract.js використовується для оптичного розпізнавання символів на обробленому за допомогою OpenCV зображенні. Ця бібліотека може розпізнавати більше 100 мов, в тому числі й українську, тому було використано пакети «eng» та «ukr» для розпізнавання англійської та української мов відповідно.

3.5 Серверна частина

Серверну частину було реалізовано за допомогою інструментів Express для Node.js. Express – це фреймворк, який дозволяє створювати сервери з відкритим прикладним програмним інтерфейсом, який містить в собі набір http методів. Загальна файлова структура серверної частини зображена в Додатку 1. Вхідним виконуваним файлом серверної частини є app.js. В даному файлі міститься код запуску http сервера та налаштування маршрутизації api (routes).

Маршрути визначають набір вузлів, за допомогою яких відбувається обмін даними з сервером за допомогою протоколу http. Серверна частина має три типи маршрутів:

- processRoutes – маршрути для відправки ФЗ або СЗ на автоматизовану обробку.
- userRoutes – маршрути для роботи з користувачем (містять вузли для авторизації користувача, для отримання інформації про кількість відправлених користувачем документів, зміни налаштувань профілю користувача, архів).
- resultRoutes – маршрути для отримання інформації про результати обробки документів (містять вузли для отримання списку оброблених документів, отримання документу за ідентифікатором, отримання часткової інформації про документ тощо).

Кожен маршрут обробляється контролером та його методами.

Контролер – це клас, який містить в собі набір методів для обробки HTTP запиту на певному маршруті. Метод контролеру отримує запит (request), перевіряє структуру запиту та наявність необхідних даних, певним чином обробляє запит, та надсилає відповідь (response) який містить результат запиту у разі успішної обробки запиту, або ж помилку і її статус у разі неуспішної обробки запиту. Для кожного типу маршрутів наявний свій контролер:

- ProcessController – обробляє маршрути processRoutes.
- ResultsController – обробляє маршрути resultRoutes.
- UserController – обробляє маршрути userRoutes.

Схема відповіді на запит має наступний вигляд:

```
{
      "success":true,
      "type": "PROCESS_CREATED",
      "data": { ... }
}
```

де success вказує на успішність виконання запиту, type – тип даних які відправляються, data – дані. Контролер оперує з даними представленими у вигляді моделей (models). Моделі описують структуру даних, набір методів для роботи з цими даними та схему запису в базу даних.

При отриманні запиту на обробку документа, сервер отримує зображення у форматі base64. Сервер декодує зображення та зберігає копію зображення у локальному сховищі. Також в БД створюється новий запис про документ який обробляється. Даний запис містить:

- внутрішній ідентифікатор документа (генерується автоматично)
- ідентифікатор користувача, якому цей документ належить
- час початку обробки документу
- посилання на копію зображення документу
- статус обробки документу
- результат обробки

Після створення запису в БД сервер надсилає запит на створення нового процесу обробки до сервісу обробки зображення. Запит містить в собі зображення і посилання на вузол, куди сервіс має відправляти повідомлення про статус обробки. Для обміну даними з сервісом обробки зображень документів використовується протокол HTTP та бібліотека для виконання HTTP запитів Axios.js.

Оскільки сервер після отримання інформації про зміну статусу обробки повинен мати можливість відправляти сповіщення до клієнту та систему обліку, постає задача у необхідності використання механізму миттєвої відправки даних до клієнту. Для вирішення даної задачі було використано механізм довгих опитувань (long-polling). Даний механізм є простим у реалізації і використовує звичайний протокол http. Робота даного механізму (рис.4) полягає в тому, що

після того як клієнт до серверу надсилає запит про наявність нових сповіщень, сервер перевіряє їх наявність, та очікує появу нових сповіщень, якщо їх немає на момент запиту, при цьому залишаючи з'єднання відкритим на певний timeout час. Якщо за час timeout сервер отримав нове сповіщення для клієнта, він відправляє відповідь до клієнта і закриває з'єднання. Якщо сервер за час timeout не отримав нових сповіщень, відправляється відповідь про відсутність нових сповіщень і закривається з'єднання. Після отримання відповіді клієнт відразу відправляє новий запит на отримання сповіщень.

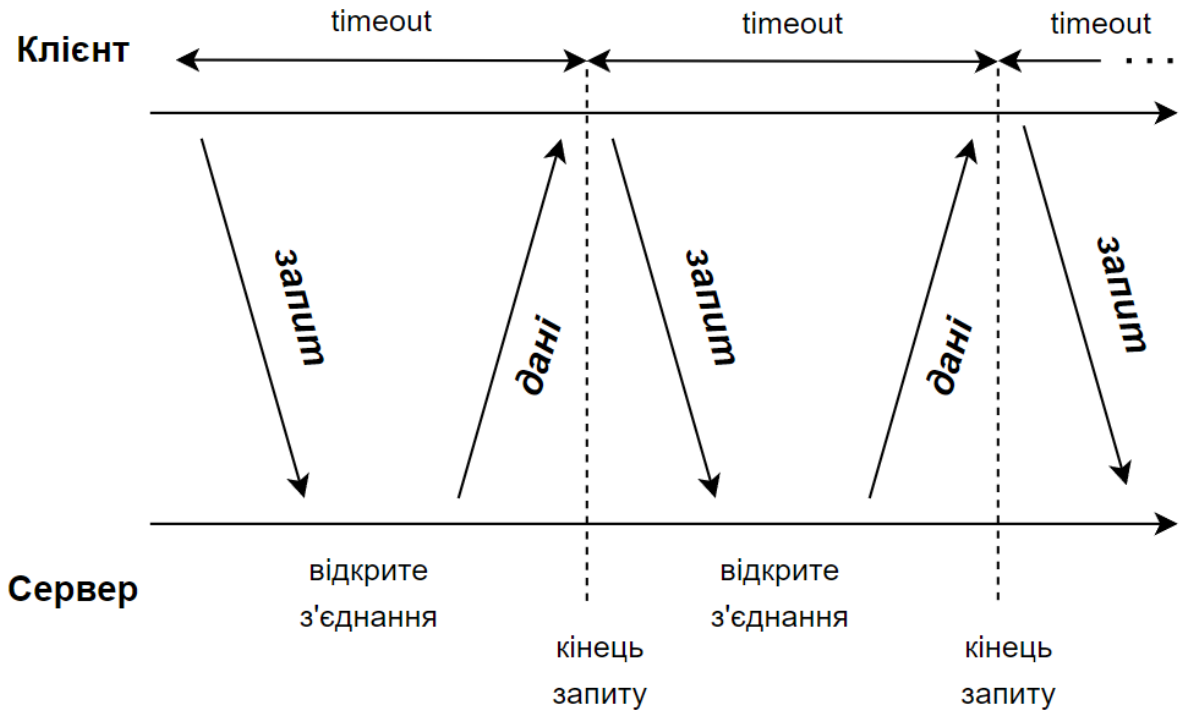


Рис.4 Схема роботи механізму довгих опитувань

Код реалізації механізму відправки сповіщень за допомогою механізму довгих опитувань надано у додатку 3.

3.6 Сервіс обробки зображення та розпізнавання тексту

Сервіс обробки зображень було реалізовано за допомогою Node.js. Для обміну даними з іншими компонентами системи сервіс має HTTP-інтерфейс. Як і сервер, сервіс обробки має налаштовані маршрути та контролери. Контролер TaskController відповідає за обробку запитів на створення нового процесу обробки документу. Запит містить в собі зображення, яке необхідно обробити, тип документу, а також посилання на вузол, куди необхідно відправляти статус про обробку документу та оброблені дані. Перед створення процесу обробки, сервіс перевіряє завантаженість ресурсів. Якщо ресурсів достатньо, створюється новий процес обробки, який виконує модуль обробки обраний за типом документу. Якщо ресурсів недостатньо, створюється завдання на обробку, яке ставить в чергу. Коли процес обробки документу завершено, перевіряється черга завдань та запускається процес для наступного завдання.

Процес обробки документу полягає в попередній обробці зображення, розпізнаванню тексту зображення та виокремлення необхідних даних. На кожному з етапів обробки відправляються дані про статус обробки.

Для виконання процесу обробки використовуються модулі обробки. Модуль обробки являє собою клас, який реалізує абстрактний клас BaseModule (Додаток 4), який містить функцію execute. На вхід функції execute подається зображення, мова тексту зображення та функція, яку буде викликано при зміні статусу обробки. Після виконання процесу обробки, функція execute повертає дані документу.

3.5 Алгоритм попередньої обробки зображення

Бібліотека Tesseract як і інші бібліотеки оптичного розпізнавання символів має проблему – ефективність розпізнавання залежить від

характеристик вхідного зображення. Без попередньої обробки зображення ефективність розпізнавання досягала близько 40-50 відсотків. Такий результат не задовольняє вимогам САОФД. Тому перед розпізнаванням тексту з зображення, потрібно його підготувати за допомогою методів OpenCV, впровадивши алгоритм попередньої обробки.

Алгоритм попередньої обробки зображення використовується для підвищення якості розпізнавання символів внаслідок покращення характеристик зображення. Алгоритм обробки включає в себе наступні кроки:

1) Отримання зображення в сірих кольорах за допомогою функції ***OpenCV.bgrToGray***.
2) Розмиття зображення за допомогою функції ***OpenCV.gussianBlur.***
3) Отримання границь елементів зображення за допомогою функції ***OpenCV.canny.***
4) Підвищення контрастності границь за допомогою функцій ***OpenCV.dilate*** та ***OpenCV.erode.***
5) Пошук контурів за допомогою функції ***OpenCV.findContours.***
6) Спрощення контурів за допомогою функції апроксимації ***OpenCV.approxPolyDP***.
7) Пошук найбільшого за площею прямокутного контуру на зображенні, якщо знайдено декілька прямокутних контурів. Якщо контурів не знайдено, даний крок пропускається.
8) Вирівнювання перспективи за допомогою функції ***OpenCV .warpPerspective.***
9) Обрізання непотрібного фону.
10) Отримання бінарного зображення за допомогою функції ***OpenCV.threshold***.

Результати зображення після обробки на кожному кроці показано в Додатку 5.

Можливий сценарій коли зображення документу повнісю покриває фон на вхідному зображенні. В такому випадку кроки 4-9 не виконуються.

3.6 Алгоритм виокремлення ідентифікаційного коду квитанції.

Після розпізнавання символів на зображенні отримуємо суцільний розпізнаний текст. Для виокремлення коду квитанції з цього тексту було використано регулярний вираз (рис. 2) та вбудовану в Javascript функцію String.match. Якщо якийсь підрядок отриманого тексту задовольняє заданому регулярному виразу, його буде повернено в якості знайденого ідентифікаційного коду. Якщо код не виявлено, користувачеві відправиться статус неуспішної операції розпізнавання.

3.7 Перевірка результатів роботи системи

Розроблену систему було протестовано в автоматичному режимі на наборі понад 400 зображень квитанцій про оплату різних банків. Набір було складено з зображень з різними характеристиками (тип зображення, якість, перспектива, наявність шумів). У 97 відсотках випадків дані про ідентифікатор квитанції було розпізнано коректно.

Загальний код розробленої системи розміщено за посиланням https://github.com/maxsalo17/docs-processing-app.

ВИСНОВКИ

В результаті випускної кваліфікаційної роботи магістра досліджено способи розробки системи автоматизованої обробки фінансових документів та розроблено таку систему і комплекс рішень на прикладі системи розпізнавання ідентифікаційного коду банківської квитанції про оплату.

Автоматизована обробка рахунків на оплату значно спрощує і оптимізує процес обробки платіжних документів та завдяки цьому дозволяє організаціям заощадити значні ресурси й підвищити продуктивність. Однак автоматизована інтелектуальна обробка документів далеко не обмежується тільки обробкою рахунків на оплату, які б складними за формою вони не були.

Ефективність оптичного розпізнавання тексту з зображення залежить від якості вхідного зображення, тому було розроблено алгоритм попередньої обробки зображення.

З огляду на цілу низку інструментів, які існують для вирішення даного завдання, важливим є створення масштабованої системи, яка легко інтегрується в уже існуючі системи, та є ефективною для вирішення поставлених задач обробки. Тому використані технології найкраще підходять для реалізації необхідного рішення, є зручними та ефективними при розробці, та легко інтегруються у існуючу екосистему інформаційних застосувань університету.

Розроблену систему протестовано на наборі понад 400 зображень документів про оплату. Ефективність розпізнавання даних документу досягає 97 відсотків. Розроблена система знаходиться в режимі тестування.

Загальний код розробленої системи є відкритим та може бути використаний іншими організаціями.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

[1] Kanevska L. Інтелектуальна обробка документів [Електронний ресурс] / Larysa Kanevska. – 2022. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/інтелектуальна-обробка-документів-розпізнавання-та-даних-kanevska-1f/.

[2] Герасимчук В. ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС (API), ЯК ІНСТРУМЕНТ РОЗРОБКИ / В.В. Герасимчук. // Житомирський державний технологічний університет. – 2019. – С. 1.

[3] Andriy Ivashchenko. REST: простым языком [Електронний ресурс] / Andriy Ivashchenko. – 2019. – Режим доступу до ресурсу: https://medium.com/@andr.ivas12/rest-простым-языком-90a0bca0bc78.

[4] HTTP [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTTP.

[5] Optical Character Recognition Systems for Different Languages with Soft Computing / C.Arindam, M. Krupa, B. Pratixa, G. Soumya K.. – Warsaw, Poland: Springer, 2022. – 247 с.

[6] Bunke H. Handbook of Character Recognition and Document Image Analysis / H. Bunke, P. Wang. – Singapore: World Scientific Publishing Company, 1997. – 852 с. – (Second edition).

[7] Афонасенко. А. В. Обзор методов распознавания структурированных символов / А. В. Афонасенко., А. И. Елизаров. – Томськ, 2008. – 83 с.

[8] Susmith R. Pre-Processing in OCR [Електронний ресурс] / Reddy Susmith. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7.

[9] Crump J. Generating an Ordered Data Set from a Text File [Електронний ресурс] / Jon Crump. – 2014. – Режим доступу до ресурсу: https://programminghistorian.org/en/lessons/generating-an-ordered-data-set-from-an-OCR-text-file.

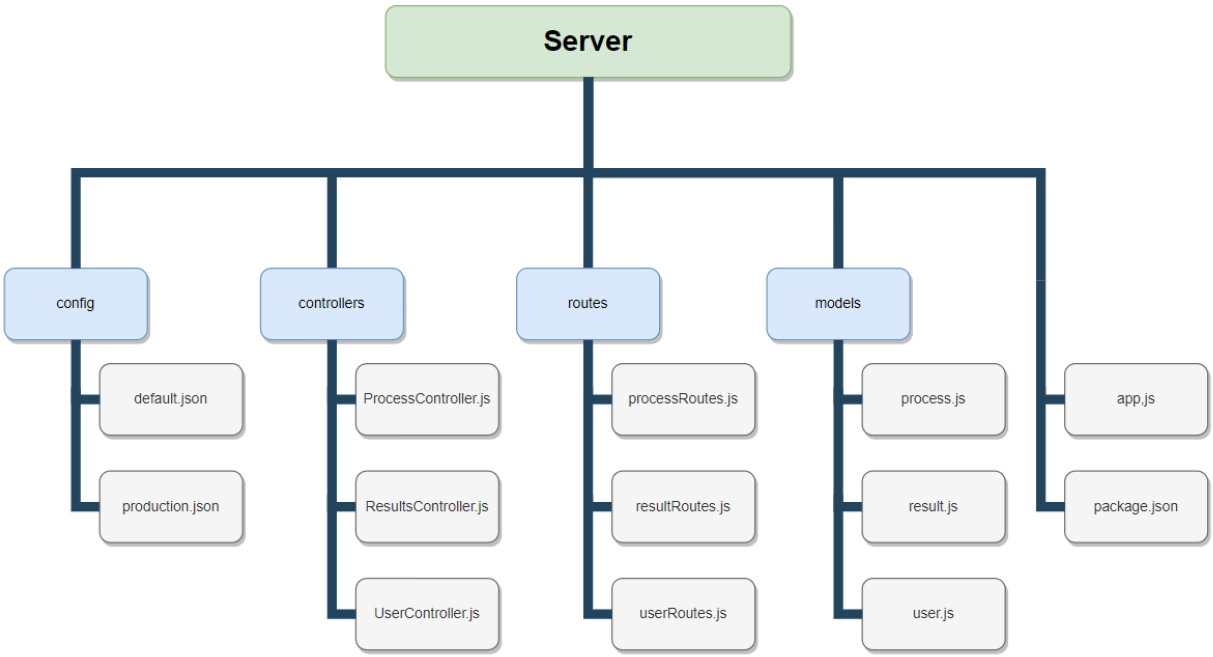
[10] Что такое Node.js? [Електронний ресурс] – Режим доступу до ресурсу: http://web.spt42.ru/index.php/chto-takoe-nodejs.

[11] Windmill E. Flutter in Action / Erik Windmill., 2019. – 368 с. – (1st edition).

[12] Rosebrock A. OCR with OpenCV, Tesseract and Python / Adrian Rosebrock., 2019.

ДОДАТКИ

Додаток 1. Файлова структура серверної частини



Додаток 2. Код файлу app.js

```
const express = require("express");
const bodyParser = require('body-parser');
const cors = require('cors')
const config = require('config')

const processRoutes = require('./routes/processRoutes')
const resultsRoutes = require('./routes/resultRoutes')
const userRoutes = require('./routes/userRoutes')

var app = express();
app.listen(config.get('serverPort'));

app.use(cors());

app.use(bodyParser.urlencoded({ limit: "50mb", extended: true, parameterLimit: 50000 }))
app.use(bodyParser.json({ limit: "50mb" }));

app.use(userRoutes, processRoutes, resultsRoutes)
```

Додаток 3. Реалізація механізму довгих опитувань

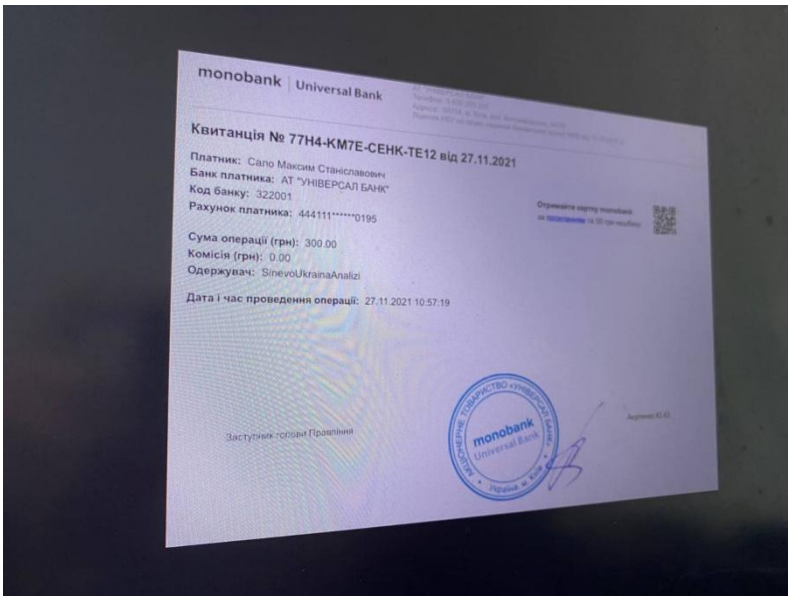
```
static async getProcessNotifications(req, res) {
    let notificationsCheckTask;
    const timeout = req.body.timeout || config.get('long_polling_timeout');
    const retryInterval = 500;
    let retry = 0;
    let user = req.body.user;
    try {
        notificationsCheckTask = setInterval(() => {
            if (retryInterval * retry > timeout) {
                res.send({
                    success: true,
                    type: 'NO_NOTIFICATION'
                });
                clearInterval(notificationsCheckTask);
            } else {
                const actionObj = this.notificationsQueue.find(element => element.id === user.id);
                if (actionObj && actionObj.action) {
                    res.send({
                        success: true,
                        type: 'NOTIFICATION_DATA',
                        data: actionObj.data
                    });
                    this.notificationsQueue = this.notificationsQueue.filter(element => element !== actionObj);
                    clearInterval(notificationsCheckTask);
                }
            }
            retry += 1;
        }, retryInterval);
    } catch (e) {
        res.status(500).json({
            success: false,
            type: 'INTERNAL_ERROR',
            error: e.message
        });
    }
}
```

Додаток 4. Абстрактний клас BaseModule для реалізації модулів обробки

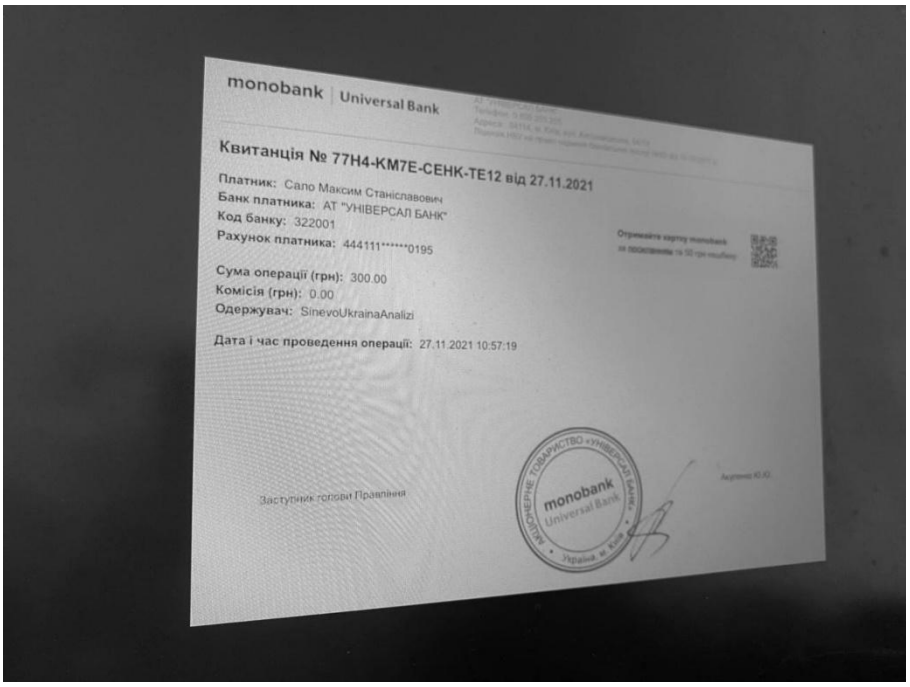
```
class BaseModule {
    async execute(image, mimetype, lang, onStatusChangedCallback) {
        throw new Error("Method 'execute()' must be implemented.");
    }

    static type;

}

module.exports = BaseModule;
```

Додаток 5. Результати обробки зображення деякими функціями бібліотеки OpenCV.

Оригінальне зображення

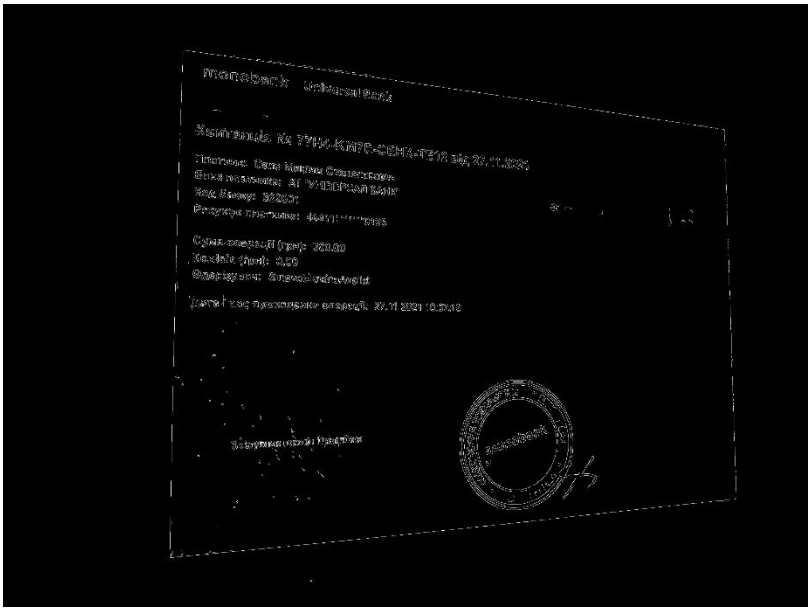


Результат функції bgrToGray

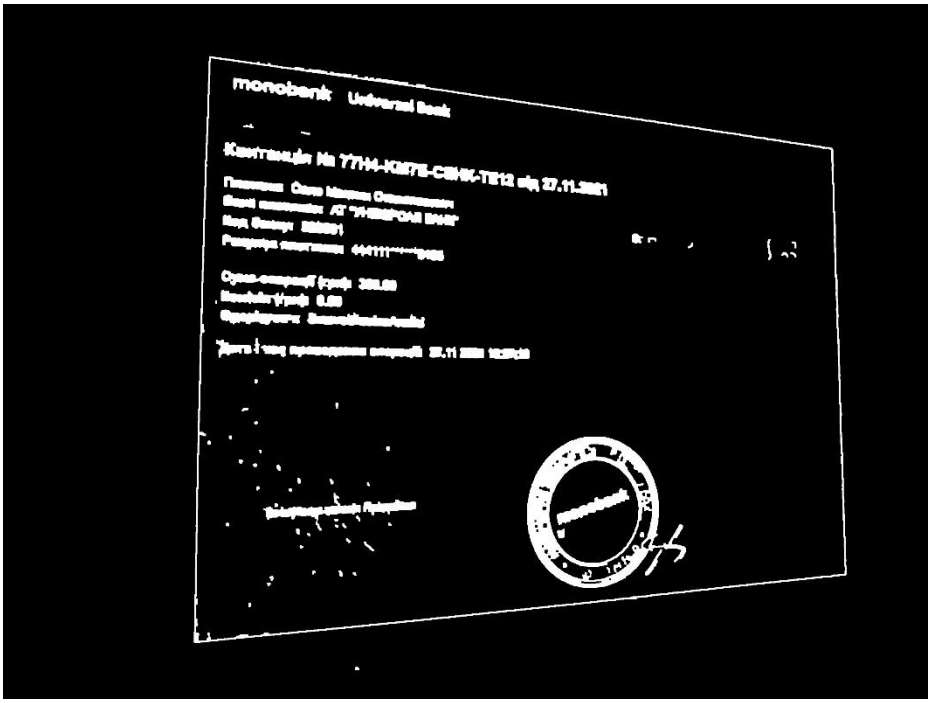


Результат функції gaussianBlur

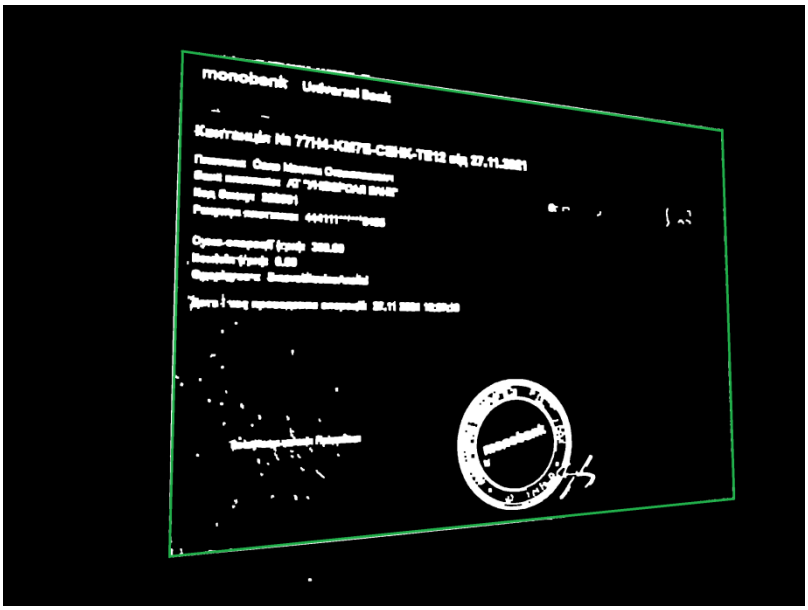
Результат роботи функцій canny



Результат роботи функцій OpenCV.dilate та OpenCV.erode

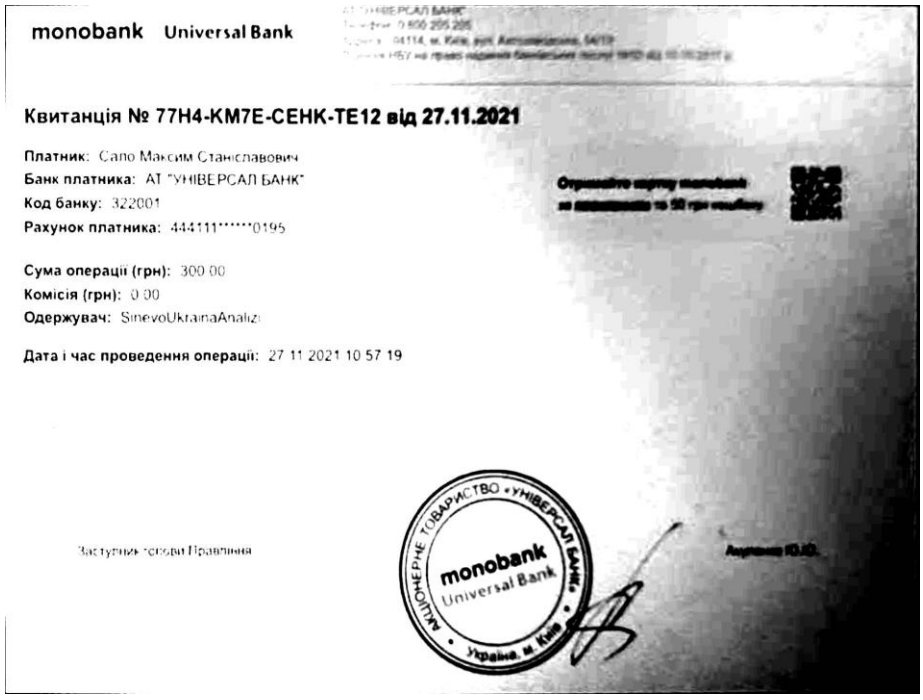


Результат роботи функцій findContours та approxPolyDP



Результат роботи функції warpPerspective

Результат роботи функції threshold



Додаток 6. Інтерфейс клієнтської частини ПЗ автоматизованої обробки фінансових документів.