ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: **«РОЗРОБКА СИСТЕМИ АВТОМАТИЗОВАНОЇ ВАЛІДАЦІЇ ОСОБИСТІ ЛЮДИНИ ДЛЯ БАНКОВОГО РИНКУ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ»**

Виконала: студентка 6 курсу, групи ПДМ–62
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Тертична Ю.М.
(прізвище та ініціали)

Керівник Негоденко О.В.
(прізвище та ініціали)

Рецензент_____
(прізвище та ініціали)

Нормоконтроль_____
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій
Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Магістр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення Негоденко О.В.
“____” _____2022року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТЦІ

Тертичній Юлії Михайлівні

(прізвище, ім’я, по батькові)

1. Тема роботи: Розробка методу автоматизованої оцінки usability сайту на основі аналізу даних користувацьких логів

Керівник роботи: Негоденко Олена Василівна, к.т.н., доц., доцент кафедри ІПЗ ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12» жовтня 2022 року №122.
2. Строк подання студентом роботи
3. Вхідні дані до роботи
Науково-технічна література з питань, пов’язаних з темою роботи.
Документ з відмітками для підтвердження верифікації.
4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Аналіз технологій розпізнавання
4.2 Основні складові у верифікації та розпізнавання документів
4.3 Викортстання мови JavaScript та API Tessaract для розробки інструменту
4.4 Розробка інструменту верифікації документів

5. Перелік демонстраційного матеріалу (назва основних слайдів)

5.1. Аналіз існуючих ІТ рішень та їх моделей

5.2. Необхідні коефіцієнти та алгоритми

5.3. Реалізована матиматична модель алгоритму

5.4. Інтерфейс користувача

5.5. Результат процесу створення системи автоматизованої валідації людини

6. Дата видачі завдання 14.10.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на магістерську роботу	14.10-16.10	Виконано
2	Аналіз методів проведення оцінки існуючих ІТ рішень та їх моделей	17.10-25.10	Виконано
3	Аналіз інструментів для оцінки автоматизованої валідації людини для банкового ринку	26.10-09.11	Виконано
4	Розробка математичної моделі та методу для системи автоматизованої валідації людини	11.11-20.11	Виконано
5	Моделювання та аналіз результатів	21.11-30.11	Виконано
6	Написання та оформлення пояснювальної записки	30.11-28.12	Виконано
7	Розробка графічних та презентаційних матеріалів	29.12-31.12	Виконано
8	Захист роботи	18.01.23	

Студентка Тертична Ю.М.

Керівник роботи Негоденко О.В.

РЕФЕРАТ

Текстова частина магістерської роботи: сторінки 45, рисунки 15, джерел 6.

Об'єкт дослідження – створення API для верифікації документів..

Предмет дослідження – розпізнавання текстів, методи верифікації документів..

Мета роботи – розробити інструменту для верифікації.

Методи дослідження – огляд методичних матеріалів, аналіз схожих систем, дослідження технологій роспізнавання.

В роботі наведено основні відомості про методи розпізнавання текстів, та інструменти котрі користуються популярностю на просторах інтернету. Детально досліджено технології розпізнавання, в тому числі існуючи методи котрі використовують на даний момент На основі результатів дослідження розроблено інструмент для верифікації документі.

Галузь використання – фінансові установи котрі мають онлайн системи і працюють з осибситими документами клієнтів.

РОСПІЗНАВАННЯ ОБРАЗІВ, ТЕКСТІВ, ВЕРИФІКАЦІЯ ДОКУМЕНТІ, ПІДТВЕРДЖЕННЯ ОСОБИ

ЗМІСТ

ВСТУП

В 21-му столітті більшість фінансових компаній переходять на роботу с клієнтами в режим онлайн, використовуючи для цього спеціально створенні WEB-ресурси. Починаючи с звичайних мікрозаймових установ закінчуючи великими торговими платформами. Процес реєстрування клієнтів має деякі правила, які змінювати не можна. Клієнт сам вводить інформацію про себе в потрібні поля, але є люди які можуть попробувати створити аккаунт на іншу особу чи ввести данні невірно, щоб такого не було використовують документ підтвердження особистості, наприклад паспорт чи водійські права. Звичайно що вони подаються в електронному вигляді як сканкопії, але з'являється проблема, при подачі інформації та документів потік їх може бути дуже великим і створювати черги в роботі онлайн, це не припустимо, а менеджери перепровіряти всі данні будуть довго. Тому з'являються методи автоматичної верифікації документів.

Об'єкт дослідження – процес вивчення методів розпізнавання тексту та образів на зображеннях.

Предметом дослідження є верифікація документів, а об'єктом дослідження – процес його розробки та запровадження.

Метою магістерської роботи є розробка плагіну для верифікації документів, що замінить собою роботу менеджерів та пришвидчить час перевірки для фінансових установ осіб.

Для досягнення мети необхідно вирішити такі основні задачі:

1. Дослідити процеси та технології існуючих методів роспізнавання;
2. Проаналізувати існуючі системи які користуються найкращою репутацією в даній галузі;
3. Дослідити технології серверних та мережевих можливостей;
4. Застосувати розроблений модуль на практиці.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність роспізнавання документів

Обчислювальні системи існують не один десяток років. Метою їх створення була можливість замінити людину, зробити за неї трудомістку роботу, яка потребує складних обчислень. Одне з таких завдань полягає в тому, як навчити комп'ютер розпізнавати образи (зокрема, який алгоритм необхідно створити, щоб комп'ютер міг сприймати зображення текстів). Розпізнавання текстів – дуже складне завдання з теоретичної та практичної точки зору. Людина, наприклад, задіює для цього всю комплекс знань та досвіду. Він визначає текст із сукупності сигналів органів чуття, виділяє кожен символ, виділяє характерні ознаки символів та на підставі свого досвіду приходить до висновку про значення символу та всього тексту в цілому.

Комп'ютер помиляється в процесі розпізнавання набагато частіше людини. Сьогодні немає абсолютно точного методу визначення тексту та символу за їхнім зображенням. Багато розроблених комерційних проекти використовують свої запатентовані методи та не можуть похвалитися ідеальним розв'язанням задачі. У багатьох випадках гарне рішення дає комплексний підхід до завдання. Саме завдання розпізнавання тексту поділяється на підзавдання: фільтрація зображення від шуму, виділення зображень символів із зображення тексту, виділення ознак символів та порівняння цих ознак з збереженими зразками. Кожне завдання містить багато варіантів рішень, з яких лише деякі є більш-менш оптимальними.

Досить розвиненим напрямом науки на сьогоднішній день є клітинні автомати. Особливий інтерес до клітинних автоматів проявляється насамперед завдяки їх простоті: на основі простих правил клітинні автомати можуть породжувати складну поведінку. Крім того, клітинні автомати - це ідеальний варіант для паралельних обчислень, вони можуть ефективно використовуватися в

багатопроцесорних системах або бути реалізовані апаратно, оскільки основна властивість їхніх правил – локальність та однорідність. Переваги клітинних автоматів можуть виявитися корисними в система розпізнавання тексту. Простота та однорідність правил може дозволити створювати складні системи на базі декількох логічних або математичних елементів і досягти результату з меншими витратами як обчислювальних ресурсів, і пам'яті. Основним напрямом досліджень, що проводяться у роботі, є визначення меж застосування клітинних автоматів у завданнях та підзавданнях, що виникають у процесі розпізнавання тексту. Для цього необхідно виконати аналіз клітинних автоматів та їх властивостей, виділити основні характеристики клітинних автоматів, необхідних у вирішенні задач розпізнавання тексту, та розробити відповідні алгоритми. У процесі дослідження необхідно створити модель та програму на її основі для реалізації розроблених ідей та алгоритмів.

Розпізнавання тексту є одним із напрямків розпізнавання образів рис 1.1. Розпізнавання образів є дуже складною завдання в теоретичному та практичному сенсах, незважаючи на те, що з нею досить легко справляються багато живих організмів і людина. Вкрай складно створити штучну систему та її технічно реалізувати для того, щоб ефективно виконувати цей процес. В даному випадку, під розпізнаванням розуміється співвідношення зображення об'єкта, його образу, набору ознак самому об'єкту. Прикладами та додатками систем розпізнавання образів можуть бути як розпізнавання тексту загалом, і окремих його символів, розпізнавання мови, людських осіб, біометричних даних людини, штрих-кодів продуктів, номерів машин та ін.

Прикладами розпізнавання тексту є: оцифрування зображень тексту (скановані книги, статті, журнали) для подальшої роботи з його цифровим аналогом, обробка анкетних бланків, розпізнавання номерів машин та написів на об'єктах і т.д. Завдання розпізнавання тексту залишається актуальною на сьогоднішній день, тому що не існує сто процентної універсальної системи з

розпізнавання тексту. Система розпізнавання тексту передбачає наявність на вході зображення із текстом (у форматі даних графічного файлу). На виході системи повинен сформуватися текст, виділений із цього зображення. Розпізнавання тексту включає наступні підзавдання і підпроцеси.

Рисунок 1.1 – Роспізнавання образів

1. Зображення, що надходить на вхід системи, повинно бути очищене від шуму і наведено до виду, що дозволяє ефективно виділяти символи та розпізнавати їх.

2. Система повинна розбити зображення на блоки тексту, ґрунтуючись на особливості його вирівнювання та розподілу по кількох колонках.

3. Зображення з текстом має бути поділено на зображення рядків, а потім на зображення символів для того, щоб надалі обробити кожен символ окремо. Після цього кроку різні системи розпізнавання працюють за своїми специфічним алгоритмам.

4. Зображення символу може оброблятись повністю, для цього воно порівнюється з наявними шаблонами. Іншим варіантом є виділення характеристик символу, що зображається: відбір характерних ознак, і класифікація даних ознак за наявними у системі критеріям. На виході четвертого кроку з'являється можливий варіант літери. Однак, зазвичай системи на цьому не зупиняються і продовжують роботу з урахуванням інших методів, уточнюючи отриманий результат.

5. Результат розпізнавання може бути незадовільним. Для отримання кращих результатів у системі може бути вбудований блок навчання. За допомогою цього блоку можна задати системі приклади зображення різних літер в даному шрифті. Після процесу навчання передбачається краща якість розпізнавання тексту.

Система розпізнавання тексту не завжди повинна слідувати всім описаним крокам, але основні дії процесу розпізнавання є загальними для будь-якого алгоритму. На рис. 1.2 показано процес аналізу документу

Існує кілька основних систем розпізнавання тексту. Всі вони є комерційними продуктами, і багато внутрішні алгоритми їх роботи приховані від загального доступу. Принцип роботи таких систем базується на кількох стратегіях, але часто алгоритм розпізнавання у найзагальнішому вигляді полягає у послідовному висуненні та перевірки гіпотез. При цьому порядок їх висування керується закладеними у програму знаннями про досліджуваний предмет та результатами перевірки попередніх гіпотез.

Нейронні мережі – це структура зв'язаних елементів, на яких задані функції перетворення сигналу, а також коефіцієнти, які можуть бути налаштовані певний характер роботи. Частина елементів структури виділені як вхідні: ними надходять сигнали ззовні, частина – вихідні: вони формують результуючі сигнали. Сигнал, який проходить через нейронну мережу, перетворюється згідно з формулами на елементах мережі і на виході формується відповідь. Нейронна мережа може служити в системі розпізнавання тексту як класифікатор.

10. Збереження документу	Документ	**1.** Виділення текстових блоків
9. Складання» документу	Блок тексту, таблиця	**2.** Виділення рядків
8. Складання» рядків	Рядок	**3.** Поділ на слова
7. Структурування гіпотез	Слово	**4.** Поділ на символи
6. Створення моделей слова	Символ	**5.** Розпізнавання символів

Рисунок 1.2 – Аналіз документу

Цей класифікатор можна навчати, налаштовуючи коефіцієнти на елементах мережі, і, таким чином, прагнути до ідеальний результат розпізнавання. Нейронні мережі з успіхом можуть застосовуватись у системах розпізнавання. тексту, але існує велика кількість недоліків, які перешкоджають їх широкому застосуванню. Для побудови мережі, яка забезпечує розпізнавання кожного символу тексту, необхідно побудувати достатньо велику мережу елементів, що призводить до великих витрат пам'яті. Ще сильніше витрачаються ресурси системи у процесі розпізнавання, оскільки функції на елементах мережі працюють із числами з плаваючою точкою. Крім цього нейронні мережі необхідно навчати на всі випадки, що, однак, не гарантує точний результат. І, нарешті, робота нейронної мережі по розпізнавання тексту багато в чому залежить від конфігурації мережі та функцій, заданих в елементах, що потребує великих зусиль для побудови ефективно працюючої мережі.

Розпізнавання тексту – це комплекс завдань, які необхідні виконати щоб одержати кінцевого результату (тексту). Чинні комерційні системи розпізнавання тексту користуються набором алгоритмів, які разом дають дуже точний результат. Одна частина системи розпізнавання може функціонувати на основі нейронної

мережі, інша частина може використовувати переваги клітинних автоматів, і, нарешті, третина системи – накопичувати статистику та її основі видавати результат. Комплекс заходів та алгоритмів, безперечно, дозволить добити більшого результату, ніж окремо взятий принцип чи алгоритм. Клітинні автомати можуть бути частиною такого комплексу. Вони мають безперечні переваги, такі як можливість паралельного обчислення, легкість та простота правил, на основі яких вони побудовані, можливість реалізації багатьох складних алгоритмів обробки зображень.

1.2 Існуючі види систем розпізнавання

1.2.1 Tesseract

Tesseract — вільна комп'ютерна програма для розпізнавання текстів, що розроблялася Hewlett-Packard з середини 1980х1900 років «пролежала на полиці». У серпні 2006 р. Google купив її та відкрив вихідні тексти під ліцензією Apache 2.0 для продовження розробки. На даний момент програма вже працює з UTF-8, підтримка мов здійснюється за допомогою додаткових модулів.

Ядро програми Tesseract було розроблено в Брістольській лабораторії Hewlett Packard і Hewlett Packard Co, Greeley штат Колорадо в 1985—1994 роках. У 1996 р. було проведено значні зміни і підготовлено порт для Windows. Потім з 1998 року — часткова міграція з C на C++. Значна частина коду спочатку написана C, але проводилися доопрацювання для сумісності з C++ компиляторами. В даний час Tesseract 3.0 збирається під Linux з GCC 2.95 і старше і під Windows з Visual C 2008 Express і старше (підтримка Visual C 6 була видалена у версії 3.0). На сьогоднішній день останньою версією є Tesseract 5.0,

заснована на LSTM.

Tesseract - це движок оптичного розпізнавання символів (OCR) з відкритим вихідним кодом, є найпопулярнішою та якіснішою OCR-бібліотекою. OCR використовує нейронні мережі для пошуку та розпізнавання тексту на зображеннях. Tesseract шукає шаблони в пікселях, літерах, словах та реченнях, використовує двоетапний підхід, який називається адаптивним розпізнаванням. Потрібен один прохід за даними для розпізнавання символів, потім другий прохід, щоб заповнити будь-які літери, в яких він не був впевнений, літерами, які, швидше за все, відповідають даному слову або контексту речення. На одному з проектів стояло завдання розпізнати чеки з фотографій. Інструмент для розпізнавання використовував Tesseract OCR. Плюсами цієї бібліотеки можна назвати навчені мовні моделі (>192), різні види розпізнавання (зображення як слово, блок тексту, вертикальний текст), легке настроювання. Оскільки Tesseract OCR написано мовою C++, був використаний сторонній wrapper с github. Відмінністю між версіями є різні навчені моделі (версія 4 має велику точність, тому ми використовували її). Нам будуть потрібні файли з даними для розпізнавання тексту, для кожної мови свій файл. Завантажити дані можна за посиланням. Чим краще якість вихідного зображення (мають значення розмір, контрастність, освітлення), тим краще виходить результат розпізнавання. Також було знайдено спосіб обробки зображення для подальшого розпізнавання шляхом використання бібліотеки OpenCV. Оскільки OpenCV написаний мовою C++, і немає оптимального рішення написаного wrapper'a, було вирішено написати власний wrapper цієї бібліотеки з необхідними функціями обробки зображення. Основною складністю є вибір значень для фільтра для коректної обробки зображення. Також є можливість знаходження контурів чеків/тексту, але не до кінця вивчено. Результат вийшов кращим (на 5-10%). Параметри:

- language - мова тексту з картинки, можна вибрати кілька шляхом їх перерахування через "+";

- pageSegmentationMode — тип розташування тексту на зображенні;
- charBlacklist — символи, які ігноруватимуться ignoring characters.

Використання тільки Tesseract дало точність ~70% при ідеальному зображенні, при поганому освітленні/якості картинки точність була ~30%.

Оскільки результат був незадовільним, було вирішено використати бібліотеку від Apple - Vision. Використовували Vision для знаходження блоків тексту, подальшого поділу зображення на окремі блоки та їхнього розпізнавання. Результат був кращим на ~5%, але й з'являлися помилки через блоки, що повторюються.

Недоліками цього рішення були:

- Швидкість роботи. Швидкість роботи зменшилася >4 разу (можливо, існує варіант розточування)
- Деякі блоки тексту розпізнавалися більше 1 разу
- Текст розпізнається праворуч наліво, через що текст із правої частини чека розпізнавався раніше, ніж текст зліва.

Ще одним із методів визначення тексту є MLKit від Google, розгорнутий на Firebase. Даний метод показав найкращі результати (~90%), але головним недоліком цього є підтримка лише латинських символів і складна обробка розділеного тексту у одному рядку (найменування — ліворуч, ціна — справа). У результаті можна сказати, що розпізнати текст на зображеннях - завдання можливе, але є деякі труднощі. Основною проблемою є якість (розмір, освітленість, контрастність) зображення, яку можна вирішити шляхом фільтрації зображення. При розпізнаванні тексту за допомогою Vision або MLKit були проблеми з неправильним порядком розпізнавання тексту, обробкою розділеного тексту.

Розпізнаний текст може бути вручну відкоригований та придатний до використання; в більшості випадків при розпізнаванні тексту з чеків підсумкова сума розпізнається добре і не потребує коригування.

Ми виділили п'ять основних етапів розпізнавання документів:

- Вирівнювання скана по вертикалі.
- Очищення скану від шумів, плям, видалення ліній, підвищення чіткості.
- Сегментація — визначення областей, де розташовані конкретні поля паспорта: прізвище, ім'я, дата видачі тощо.
- Розпізнавання символів – класичний optical character recognition, знайомий нам ще з Fine Reader.
- Постпроцессинг - звірка розпізнаних П.І.Б. з частотним словником та перевірка числових полів regexp`ами.

Вирівнювання скана

Отже, отримали зображення другої та третьої сторінок паспорта. Насамперед треба зменшити його до стандартного розміру скана (приблизно 1060x1500 px) — щоб подальша обробка була занадто довгою. Потім нормалізуємо картинку. Паспорти часто знімають під кутом — а для розпізнавання вони мають бути вертикально. YOLOv3 проводить вектор по центру зображення та визначає межі сторінок паспорта (чорна лінія та блакитний та зелений прямокутники відповідно). Потім помічаємо центри кожної області та з'єднуємо їх (червона лінія). Так знаходимо кут між отриманими векторами та за допомогою OpenCV повертаємо зображення на цю величину.

Очищення

Тепер скан вирівняний по вертикалі, але в ньому залишаються шуми: смуги від принтера або засвіти від сканера. Якщо зображення робили за допомогою смартфона, можуть бути і сонячні відблиски або трапеції - за рахунок того, що паспорт виявився напівзігнутим при зйомці.

Для очищення використовуємо стандартні алгоритми OpenCV: видалення шумів (non-local means denoising), зменшення розмірів контурів (erosion),

збільшення контрастності та чіткості тощо.

Окрема проблема - на деяких сканах при обробці з'являються вертикальні чорні лінії завтовшки 1-2 пікселя, які можуть впливати на розпізнавання.

Їх позбуваємося в кілька етапів. Спершу трессхолдимо зображення за методом Otsu, який встановлює пороги для білого та чорного кольорів. Конвертуємо цими порогами в бінарне чорне і біле зображення. Потім інвертуємо кольори, отримуючи зображення з білою смугою.

Це потрібно для того, щоб зробити нашу білу лінію гарантовано безперервною. Процес не шкодить основному зображенню.

Далі шукаємо усі білі лінії, які задовольняють наступним вимогам:

- перетинають область із даними паспорта;
- займають щонайменше 73% довжини зображення;
- не товщі двох пікселів (якщо видалимо товстіші лінії, то не зможемо відновити текст, через який вони проходять).

Сегментація

Сегментація - визначення сегментів скана, в яких містяться потрібні нам дані і немає зайвого.

Власноруч розмітили п'ять сотень паспортів: виділили поля з даними та вказали, у яких полях, які дані лежать. Розмічений таким чином датасет подавали в нейросітку, навчали її, і дивилися, як вона справляється розпізнаванням тестової пачки сканів паспортів.

Довелося чимало помучитися з експериментами з розмітки датасету, щоб зрозуміти, що варто виділяти в одне поле, а що — ні, щоб отримати мінімум помилок на тестовому датасеті.

Розпізнавання символів

Вже очистили зображення, знайшли всі потрібні поля: П. І. О., серія, номер, ким виданий і таке інше. Тепер зменшуємо товщину шрифту і, якщо це необхідно, збільшуємо чіткість та контраст тексту.

Навіщо це потрібно? По-перше, шрифти в паспортах різні (одним шрифтом написані тільки серія і номер і найчастіше МНС — зона машиночитаної внизу сторінки). Занадто жирний шрифт нейромережа може не розпізнати і, наприклад, видати умовну вісімку замість трійки. По-друге, яскравість і чіткість у більшості полів різні, і світло-сірий текст на світло-сірому тлі так само плутає алгоритм.

Зрештою, завантажуємо все в Tesseract, який працює з кожним рядком окремо (це підвищує якість розпізнавання) і робить нам класичний OCR – перетворює зображення на текст.

Постпроцесинг

Після розпізнавання тексту підключаються регулярні вирази, які прибирають зайві знаки, замінюють "про" на "0" і "з" на "3" у числових полях.

І на фінальному етапі використовуємо частотний словник усіх можливих П. І. О. Як він працює? На основі клієнтської бази банку створили три незв'язані між собою файли: з іменами, прізвищами, по батькові та їх частотою серед клієнтів. З них зробили довідники, за якими перевіряємо, чи є у П. І. Б. помилки і при необхідності змінюємо дані на більш ймовірні. За роботу зі словником відповідає бібліотека SymSpell. Словник покращив якість розпізнавання приблизно на 10%

Нарешті, звіряємо всі розпізнані дані з зоною машиночитання, всередині якої закодовані всі дані паспорта. Якщо розпізнані дані збіглися, це успіх і означає, що автоматична авторизація сталася.

Результати

Кількість автоматичних авторизацій заявок на кредит зросла з 31% до 82%. Час розпізнавання знизився з 11 до 5 секунд без використання обчислень на відеокарті.

Крім того, усунули залежність від вендора щодо розпізнавання — і тепер можемо самі писати сервіси розпізнавання інших документів, змінювати

компоненти системи та робити точкові оптимізації.

У планах — розпізнавання документів формату А4 із дрібним шрифтом та розширення лінійки шаблонів внутрішньобанківського розпізнавання. А також словник ділової лексики, що часто використовується у договорах.

1.2.2 Google Cloud Vision

Cloud Vision API ідея платформи – надати технології Computer Vision, у яких Google є безумовним лідером як сервіс. Кілька років тому під кожне завдання існувала своя технологія. Не можна було взяти щось спільне і досягти, щоб алгоритм вирішував усе.

- Label Detection — детектування класу якого відноситься зображення, детектування того, що зображено на зображенні: котики, собачки, слоники, безтурботність, і.т.д.
- OCR — розпізнавання тексту
- Explicit Content Detection - детектування будь-якого поганого контенту: вайп неграми та інші страхи життя.
- Facial Detection - детектування осіб, рис осіб, особливих точок на обличчях
- Landmark Detection - детектування геолокації по фото
- Logo Detection - детектування символів та значків

Так чи інакше, цією функцією від Google користуються всі. Пошук за картинкою на google.com використовує саме її. По суті, Label Detection дає зображення характеристики. Характеристика може бути об'єкт, зображений на фото. Можливо стиль фотографії, наприклад "Макро", "Портрет", "Чорно-біле". А може, щось дуже спільне: «ботанічний об'єкт», «атмосферний феномен».

Окрім пошуку ця функція може вирішити завдання:

- Сортування бази зображень
- Підписи тегів до якогось фотобанку
- Аналіз інтересів за фотографіями

Аналоги. Але, як не дивно, саме в цьому напрямку у Google є безліч конкурентів:

- Microsoft. Вміє досить непогано описувати, що відбувається на зображенні, а не тільки його складову частину. Немає онлайн-демки, щоб порівняти.
- IBM куди більш бідна і погана розпізнавала.
- Cloud Sight - каламутне розводилове. Типово прикидаються, що вони автоматична система, яка у 100% правильно розпізнає. Насправді сидять індуси. Бажають по 50 доларів за 800 зображень. У мене розпізнавало дуже погано. Але, може, просто всі вийшли курити.
- Clarifai. Працює офігенно. Я навіть не повірив. Але розпізнає і підписує найкраще протягом 2-3 секунд. Іноді, щоправда, Гугл перемагав

Є ще кілька дрібніших гравців із гірше працюючим розпізнаванням

Є навчені на ImageNet відкриті сітки, які можна налаштувати. Дешево і сердито. Але працюватиме не дуже.

Facial Detection

Розпізнавання особи. Що вміє гугл:

- Знайти обличчя
- Знайти спеціальні точки
- Визначити вираз обличчя

Що він не вміє, але вміють конкуренти:

- Визначити стать

- Визначити вік

Порівняти дві особи та прийняти рішення однакові вони чи ні.

1.2.2 ABBYY FineReader

У нас є сторінка з текстом, розбираємо її на текстові блоки, потім блоки розбираємо на окремі рядки, рядки на слова, слова на літери, літери розпізнаємо, далі по ланцюжку збираємо все назад у текст сторінки.Про рівень від документа до рядка тексту поговоримо наступного разу. Це велика система, де є багато своїх складнощів. Як деяке введення, мабуть, можна залишити тут ось таку ілюстрацію до алгоритму виділення рядків (рис 1.3). Невелике попередження: система розпізнавання FineReader дуже велика і постійно допрацьовується протягом багатьох років. Тому до написаного далі рекомендуємо ставитись як до якоїсь дуже узагальненої теорії, яка стоїть за практичною системою. Невелике попередження: система розпізнавання FineReader дуже велика і постійно допрацьовується протягом багатьох років.

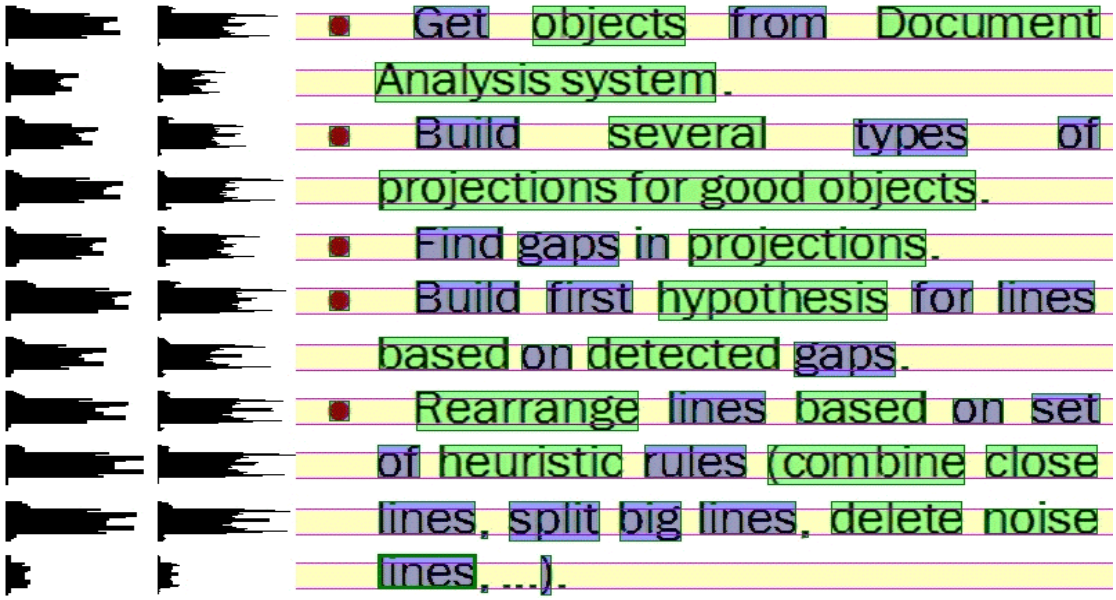


Рисунок 1.3 – Виділеня рядків

Тому до написаного далі рекомендуємо ставитись як до якоїсь дуже узагальненої теорії, яка стоїть за практичною системою. Тобто спільні ідеї та напрями в технології приблизно схожі на правду, але щоб зрозуміти до дрібниць, що ж там на практиці відбувається, краще не читати цю статтю, а працювати над розробкою цієї системи.

Граф лінійного поділу

Отже, маємо чорно-біле зображення рядка тексту. Насправді зображення, звичайно, сіре або кольорове, а чорно-білим стає після бінаризації (про бінаризацію теж потрібно писати окрему статтю, а поки що частково може допомогти ось це). Так от, нехай є чорно-біле зображення рядка тексту. Потрібно його поділити на слова, а слова на символи для розпізнавання. Базова ідея, як завжди, очевидна – шукаємо на зображенні рядка вертикальні білі просвіти, а далі кластеризуємо їх за шириною: широкі просвіти – це прогалини між словами, вузькі – між символами (рис 1.4).

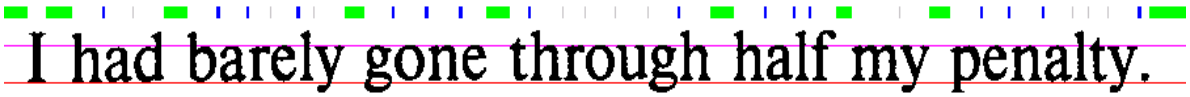


Рисунок 1.4 – Вигляд тексту при роспізнавані

Ідея чудова, але в реальному житті ширина прогалин може бути дуже неоднозначним показником, наприклад, для тексту з нахилом або невдалого поєднання символів або тексту, що злипся (рис 1.5).

Рисунок 1.5 – Вигляд не стандартного тесту при роспізнаванні

Вирішень у проблеми, загалом, два. Рішення перше – вважати якусь «видиму» ширину просвітів. Людина може практично будь-який текст, навіть незнайомою мовою, точно поділити на слова, а слова — на символи. Це тому, що мозок фіксує не вертикальне відстань між символами, а певний видимий обсяг порожнього простору з-поміж них. Рішення хороше, його, звичайно, використовуємо, тільки працює воно не завжди. Наприклад, текст може бути пошкоджений при скануванні і деякі необхідні просвіти можуть зменшитися або, навпаки, сильно збільшитися.

Це призводить до другого рішення – графа лінійного поділу. Ідея в наступному – якщо є кілька варіантів, де поділити рядок на слова, а слова на літери, то відзначимо всі можливі точки поділу, які ми змогли вигадати. Шматок зображення між двома зазначеними точками вважатимемо кандидатом літери (або слова). Варіант лінійного графа може бути простим, якщо текст хороший і немає проблем з визначенням точок поділу або складним, якщо зображення було погане (рис. 1.6, рис. 1.7).

Рисунок 1.6 – Лінійний метод

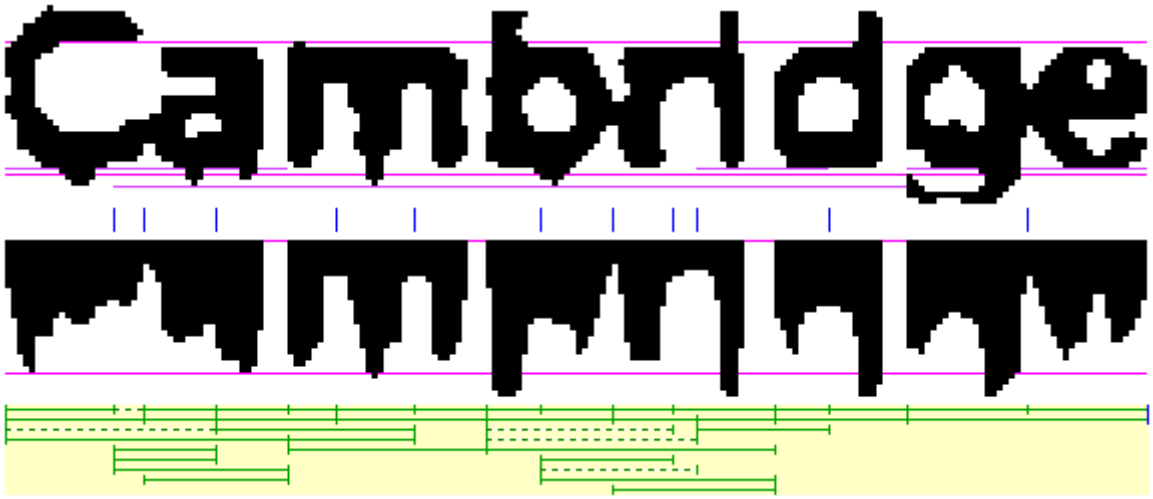


Рисунок 1.7 – Лінійний метод

Тепер завдання. Є безліч вершин графа, потрібно знайти шлях від першої вершини до останньої, що проходить через якусь кількість проміжних вершин (не обов'язково все) з найкращою якістю. Починаємо думати, що це нагадує. Згадуємо курс оптимального управління з інституту, розуміємо, що це підозріло схоже на завдання динамічного програмування.

Для кожної дуги у графі потрібно визначити її якість. Якщопрацюємо з графом лінійного поділу слова на символи, то кожна дуга у нас символ. В якості дуги використовуємо впевненість розпізнавання символу (як її порахувати — поговоримо пізніше). А якщо працюємо з ГЛД на рівні рядка, то кожна дуга цього ГЛД є варіантом розпізнавання слова, який у свою чергу був отриманий із символьного графа. Тобто нам потрібно вміти оцінювати загальну якість повного шляху у графі лінійного поділу.

Якість повного шляху у графі визначатимемо як суму якості всіх дуг МІНУС штраф за весь варіант. Чому саме мінус? Це дає нам можливість швидко оцінити максимально можливу якість варіанта шляху за сумою якості дуг цього шляху, а це означає, що більшість варіантів відсікатимемо ще до підрахунку загальної якості варіанту.

Таким чином, для ГЛД приходимо до стандартного алгоритму динамічного програмування - знаходимо точки лінійного поділу, будуємо шлях від початку до кінця по дугах з найбільшою якістю, вираховуємо підсумкову вартість збудованого варіанта. А далі перебираємо шляхи в ГЛД у порядку зменшення сумарної якості елементів з постійним оновленням знайденого кращого варіанту, доки не зрозуміємо, що всі необроблені варіанти явно гірші, ніж поточний варіант.

Гіпотези зображення

Перш ніж спустимося на рівень розпізнавання окремих слів, маємо ще одну тему, яка не обговорювалася – гіпотези зображення фрагмента.Ідея в наступному – маємо зображення тексту, з яким збираємося працювати. Дуже хочеться всі зображення обробляти однаковим чином, але правда в тому, що в реальному світі зображення всі різні - вони можуть бути отримані з різних джерел, можуть бути різної якості, вони можуть бути по-різному відскановані.

З одного боку, здається, що різноманітність можливих спотворень має бути дуже великою, але якщо почати розбиратися, виявляється лише обмежений набір можливих спотворень. Тому використовуємо систему гіпотез тексту.Маємо певний набір можливих гіпотез проблемного тексту. Для кожної гіпотези слід визначити:

- Швидкий спосіб з'ясувати, чи застосовна дана гіпотеза до поточного зображення, причому зробити це тільки на основі характеристик зображення, до розпізнавання.
- Метод виправлення на зображенні проблем конкретної гіпотези.
- Критерій якості правильності вибору гіпотези за підсумками розпізнавання зображення плюс, можливо, рекомендації для наступних гіпотез.

На зображенні вище можна побачити гіпотези для різної бінаризації та контрастності вихідного зображення (рис 1.8).

Рисунок 1.8 – Гіпотези зображень

В результаті опрацювання гіпотез виглядає так:

1. По зображенню згенерувати найбільш відповідну гіпотезу.
2. Виправити викривлення від обраної гіпотези.
3. Розпізнати отримане зображення.
4. Оцінити якість розпізнавання.
5. Якщо якість розпізнавання покращилася, то оцінити, чи потрібно застосовувати нові гіпотези до зміненого зображення.
6. Якщо якість погіршилася, то повернутися до вихідного зображення та спробувати застосувати до нього якусь іншу гіпотезу.

1.3 Технології роспізнавання

Технологія машинних методів розпізнавання застосовується у низці різних сфер людської діяльності — соціології, картографії, інженерії, автоматизації виробництва, системах безпеки, документообігу. Як приклади можна навести перепис населення, обробку знімків з космосу до створення навігаційних карт і прогнозів погоди, розпізнавання штампованих деталей на конвеєрі їхнього подальшого сортування роботизованими комплексами, перевірку справжності різних документів, технології розпізнавання особи, промови, відбитків пальців тощо.

У цьому різноманітті, найбільш затребуваної, отже, і успішної з комерційної погляду є технологія оптичного розпізнавання символів (optical character recognition, OCR). Поява сканерів дозволило швидко отримувати зображення малюнків та тексту. Але щоб текст можна було редагувати, вносити виправлення, зображення необхідно розпізнати, перевести в текстовий формат,

якщо завгодно - прочитати.

Вирішення цієї простої, на перший погляд, завдання ускладнюється різноманіттям шрифтів різного накреслення, мов, ну і, звичайно, недостатньою розбірливістю вихідних документів (факси, неякісні копії, документи, що вигоріли на сонці, розмиті фотографії, різні позначки в тексті, сліди від кави і т.п. .д.).

Більше того, для забезпечення високої точності розпізнавання документа програма повинна не лише банально передавати вихідний текст, а й враховувати логічну структуру документа з усіма вихідними елементами форматування, включаючи таблиці, зображення та підписи до них, колонтитули, перетікання тексту, вихідні шрифти та.

Декілька прикладів. Читаючи текст, людина навіть не уявляє, яку роботу робить його мозок, щоб перетворити окремі літери та їх комбінації, слова, на цілісний текст. Для штучного інтелекту літери — просто символи, мають певне зображення, що підлягає звірці із запрограмованим зразком чи прогнозом програми, заснованому деяких пропорціях введеного символа. Людина, що переглядає таблицю, у якій проведено кордону і не позначені осередки, однаково сприймає її як таблицю. Для програми всі ці правила мають бути формалізовані.

Тут доречно згадати і технології розпізнавання форм. Вона покликана автоматизувати введення типових документів, що ґрунтуються на шаблонах та заповнюються як «від руки» (так зване рукодрукове введення), так і шляхом машинного набору тексту (анкети, бланки звітності, тести).

Тут перед програмою постають додаткові завдання: розпізнати рукодрукарний або машинний символ (тобто проаналізувати набагато більше можливих відхилень кожного символу від запрограмованих еталонів) і визначити тип і назву поля, якому відповідає аналізований текст. Наприклад, у полі анкети «Рік народження» людина пише цифри, а програма повинна відрізнити цифру «0» від великої літери «О», тому що поле «Рік народження» є

числовим. Результатом розпізнавання такої програми стає не текст, а певний запис у відповідному полі бази даних. А результатом для користувача є заощаджений час на обробку величезної кількості однотипних документів.

Програмне забезпечення OCR зазвичай працює з великим растровим зображенням сторінки, отриманим у результаті сканування. Зображення зі стандартним ступенем роздільної здатності виходять при скануванні з точністю 300-300 пікселів на дюйм. Зображення паперового аркуша формату А4 при цьому роздільній здатності займає близько 1 Мб пам'яті. Зображення з більшою роздільною здатністю обробляються програмою довше і вимагають більшого обсягу оперативної пам'яті. При цьому далеко не завжди збільшення дозволу тягне за собою поліпшення якості розпізнавання.

Більшість систем мають шаблони, створені для різних рис тих чи інших символів. Програма визначає основне накреслення символів, що використовується в оброблюваному документі, і шукає відповідні символи з цим накресленням. Існують multifont-(шрифтові) та omnifont-(шрифтонезалежні) алгоритми цього процесу.

У випадку multifont растрове зображення накладається на шаблон, після чого вибирається найбільш схожий з шаблонів, що існують на досліджуване зображення.

Омніфонтові алгоритми ідентифікують символ за закладеними у програму правилами його написання. Обидва ці алгоритми не гарантують високу надійність розпізнавання, проте дозволяють зробити припущення про належність цього символу.

Тільки шаблонний опис може застосовуватися переважно для розпізнавання друкованих символів, причому якісно надрукованих. Цікавим є той факт, що рукописні шрифти теж розпізнаються із застосуванням шаблонів, але одночасно зі структурним підходом.

Алгоритми цього підходу були розроблені ABBYY для розпізнавання

текстів низької якості. Програма зберігає інформацію не про правила написання символу, а про наявність у ньому структурних елементів (кілець, дуг, кутів, відрізків та крапок).

Цей метод дозволяє виділяти елементи на спотворених зображеннях, краще знаходить нові шрифти та елементи форматування тексту (курсив, жирне зображення, верхній і нижній індекси).

Інший підхід – контекстне розпізнавання. Людина здатна швидко розрізнити на папері «В» і «Е» над останню завдяки тому, що він знає контекст слова, у якому зустрічаються ці букви. У OCR-системі за допомогою алгоритмів розпізнавання і корекції помилок служать вбудовані словники. Які, втім, потребують постійного апдейту.

Сучасні OCR-програми поєднують різні підходи, застосовуючи в залежності від типу вихідного документа.

Таким чином, можемо зробити проміжний висновок, що алгоритм розпізнавання є процесом послідовного висування та перевірки програмою цілого ряду гіпотез, заснованих на закладених у програму шаблонах і знаннях.

Користувач поміщає документ у сканер, натискає кнопку, і через деякий час до комп'ютера надходить електронне зображення, "фотографія" сторінки. На ній є всі особливості оригіналу, аж до найдрібніших подробиць. Це зображення містить всю необхідну для системи OCR інформацію про вихідний документ.

А далі вмикається штучний інтелект, який використовує машинні підходи, описані вище.

Проте найкращі у світі системи оптичного розпізнавання — найточніші, найшвидші та найнадійніші — конструює жива природа. У тому числі й система, що вірою і правдою служить кожному з нас, наш внутрішній «розпізнавач». Механізми, що дозволяють людині безпомилково пізнавати побачені предмети, поки не досліджені досконало, проте їх базові принципи добре вивчені. Таких нараховують три.

Принцип цілісності (integrity), за яким споглядається об'єкт сприймається як ціле, що з пов'язаних елементів. Зв'язок елементів виявляється у просторових відносинах з-поміж них, і самі частини отримують тлумачення лише у складі передбачуваного цілого, тобто у межах гіпотези про об'єкт. Приклад: бачимо зображення деревоподібної структури. Розпочато розпізнавання. Висуваються гіпотези: це або малюнок дерева, і тоді "гілки" структури відповідають гілкам, або схема автобусних маршрутів, де "гілки" позначають шляхи автобусів з різними номерами, або це карта річкової заплави, а "гілки" - русла річок і струмків.

Принцип цілеспрямованості (purposefulness) формулюється просто: будь-яка інтерпретація даних має певну мету. Відповідно до цього принципу, розпізнавання є процес висування гіпотез про цілий об'єкт і цілеспрямованої їх перевірки. Приклад (продовження): якщо зображення, що спостерігаємо, — схема маршрутів, то на «гілках» повинні бути позначені зупинки. Якщо зображення — карта заплави, то мають бути назви річок і струмків, а також масштаб. Якщо ж це малюнок дерева, на «гілках» ймовірно наявність листя, а біля основи – зображення трави чи землі. Перевірка: позначень зупинок немає, листя і трави немає, кожна «гілка» має назви, внизу проставлено масштаб. Підтверджено гіпотезу: це карта річкової заплави, а «гілки» відповідають руслам. Розпізнавання закінчено.

Принцип адаптивності (adaptability) має на увазі здатність системи до самонавчання. Отримана при розпізнаванні інформація впорядковується, зберігається та використовується згодом під час вирішення аналогічних завдань. Перевага самонавчання систем полягає в здатності спрямовувати шлях логічних міркувань, спираючись на раніше накопичені знання. Приклад: бачимо нове зображення деревоподібної структури, внизу проставлено масштаб. Інформація: минулого разу таке зображення виявилося картою, тому, перш ніж висувати інші гіпотези, слід перевірити наявність назв річок. Перевірка: назви виявлено.

Розпізнавання закінчено.

Замість повних назв цих принципів часто використовують абревіатуру IPA, складену з перших літер відповідних англійських слів. Переваги системи розпізнавання, що працює відповідно до принципів IPA, очевидні навіть фахівцю; саме вони здатні забезпечити максимально гнучку та осмислену поведінку системи. Цілком можна порівняти з тим, що демонструють живі «розпізнавачі», створені природою.

Відповідно до вищеописаних принципів на всіх етапах обробки документа діє ABBYY FineReader — безсумнівно найпоширеніша на пострадянському просторі OCR-система.

Наприклад, на етапі розпізнавання символів зображення, згідно з принципом цілісності, буде інтерпретовано як якийсь об'єкт тільки в тому випадку, якщо на ньому є всі структурні частини цього об'єкта, і ці частини знаходяться у відповідних відносинах. Інакше кажучи,

ABBYY FineReader не намагається ухвалювати рішення, перебираючи тисячі еталонів у пошуках найбільш відповідного. Натомість висувається ряд гіпотез щодо того, на що схоже виявлене зображення, потім кожна гіпотеза цілеспрямовано перевіряється.

Причому, на відміну від конкурентів, «Файн» набагато краще справляється саме з форматуванням тексту та всього документа, а не лише «голого» тексту. Як і слід чинити, виходячи з принципу цілеспрямованості. Причому перевіряти, чи правильна гіпотеза, система буде, спираючись на накопичені раніше відомості про можливі накреслення символу в розпізнаваному документі. У повній відповідності до принципу адаптивності.

Отже, дійшли початку процесу розпізнавання. За допомогою OCR-програми комп'ютер зможе «прочитати» на відсканованій сторінці текст, відокремивши його від ілюстрацій та інших елементів оформлення, знайти таблиці та розібратися у їхньому вмісті. А потім знову скомпонувати все це в

зручному, придатному для редагування вигляді, знову відтворивши зовнішній вигляд сторінки.

Для введення великих обсягів застосовується потокове сканування документів спеціальних промислових документних сканерах. Обробка в таких системах проводиться у напівавтоматичному режимі з великою продуктивністю. Поточне сканування документів оптимальне для створення електронного архіву великого обсягу однотипної інформації (бухгалтерської документації, звітів, висновків, наукових праць тощо). Потокове сканування застосовується для оцифрування: бухгалтерських та фінансових документів, договірних документів, юридичних документів, архівних документів, каталогів бібліотек та ін.

Кошти Image-processing застосовуються при автоматичному введенні даних в інформаційні системи з будь-яких видів документів (що засвідчують особу, бухгалтерських, юридичних тощо) для створення електронних архівів з можливістю швидкого пошуку потрібних документів, при обробці великих масивів даних (перепис населення, єдиний) держекзамен тощо), а також для переведення відсканованих документів, зображень та PDF-файлів у формати, що редагуються. впровадження сучасних засобів потокового введення дозволяє знизити витрати на обробку документів більш ніж на 50%, досягти збільшення швидкості введення в інформаційні системи у 3—10 разів, забезпечити підвищення зручності та якості роботи з даними (високий рівень безпеки конфіденційних даних, скорочення кількості помилок, пов'язаних з з людським фактором при введенні даних), оптимізувати бізнес-процеси за рахунок автоматизації рутинної функції введення даних та звільнення часу працівників на вирішення профільних завдань. При цьому середня окупність запровадження становить від трьох місяців до одного року.

Головними споживачами Image-processing у світі є великі організації (трохи більше половини обсягу ринку в грошах), на частку середніх підприємств припадає близько третини, решта — малий бізнес.

2 ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА

2.1 Постановка задачі

З розвитком технологій зявилася потреба в автоматизації бізнес процесів, в тому числі документообігу. За рахунок чого зявилися нові методи обробки документів та пітвердженя їх справжності і особистості їх володаря. Також розвивається штучний інтелект який допомагає в прискоренні створення потрібного ПО для робдоти з ними.

Потрібно розробити систему верифікації документів, що можна буде легко інтегрувати в будь-яке WEB-серидовище, без додаткового втручання спеціалістів та с простотою використання його, тобто створення API.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- простий в інтеграції;
- високий коефіцієнт достовірності;

можливість інтегрувати с різними типами документів та мовами;

2.2 Вибір технологій та серверних елементів. Обґрунтування вибору

Программа буде розроблена на мові програмування JavaScript, використання додаткових бібліотек JQuery, API Tessaract та використовувати сервер VPS. Вибір інструментів впав на текстовий редактор Sublime Text 3, та файлообмінник FileZila.

JavaScript - це мультипарадигменна мова програмування, яка зазвичай застосовується як вбудований інструмент для програмного доступу до різних об'єктів додатків. З погляду веб-розробки, без знання цієї технології неможливо

займатися створенням сучасних інтерактивних сайтів. Мова JS – це те, що «оживляє» розмітку сторінок (HTML) та функціонал користувача (CMS) сайтів. За допомогою цієї мови реалізується можливість реакції сторінки або її окремих елементів на дії відвідувача. Наразі JavaScript є базовою мовою програмування для браузерів. Він повністю сумісний із операційними системами Windows, Linux, Mac OS, а також усіма популярними мобільними платформами.

Тепер потрібно розібратися з тим, що таке JavaScript із погляду функціонування. Всі дії користувача у вікні браузера створюють події, а програмування на JS дозволяє обробляти їх певним чином.

Стандартний алгоритм роботи виглядає так:

- Користувач виконав певну дію.
- Браузер визначив подію.
- Активується JS-код.
- На сторінці відбувається задана зміна.

Завдання програміста полягає в тому, щоб створити оброблювачі для всіх подій, на які повинен реагувати сайт при взаємодії з користувачем. Для наочності наведемо два приклади типових сценаріїв.

- Користувач натискає ліву кнопку миші.
- Браузер фіксує подію onclick.
- Активується функція changePhoto.
- У вікні перегляду фотографій змінюється зображення.

Якщо обробник не впроваджено в код, алгоритм роботи буде приблизно наступним:

- Користувач робить клік.
- Браузер фіксує подію onkeydown.
- Для його обробки немає спеціального JS-коду.
- Після кліку нічого не відбувається.

Слід зазначити, що програмування не завжди прив'язується до активних

дій користувача. Наприклад, JavaScript-код може спрацьовувати за повного завантаження сторінки або після певного часу знаходження на сайті. Ці можливості активно використовуються для створення спливаючих Pop-up елементів та чатів.

У чистому вигляді, тобто без спеціальних надбудов, мова JS може функціонувати тільки в рамках браузера. Але навіть тут є обмеження щодо вимог безпеки. Наприклад, можливості JavaScript дозволяють закрити лише ту вкладку, яка була створена ним. Загалом, ця мова програмування створена для браузерів та інтернету, а за їх межами вона і не повинна була використовуватися.

jQuery — популярна JavaScript-бібліотека з відкритим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних вебсторінок.

Tesseract — вільна комп'ютерна програма для розпізнавання текстів, що розроблялася Hewlett-Packard з середини 1980х1900 років «пролежала на полиці». У серпні 2006 р. Google купив її та відкрив вихідні тексти під ліцензією Apache 2.0[2] для продовження розробки. На даний момент програма вже

працює з UTF-8, підтримка мов здійснюється за допомогою додаткових модулів.

Ядро програми Tesseract було розроблено в Брістольській лабораторії Hewlett Packard і Hewlett Packard Co, Greeley штат Колорадо в 1985—1994 роках. У 1996 р. було проведено значні зміни і підготовлено порт для Windows. Потім з 1998 року — часткова міграція з C на C++. Значна частина коду спочатку написана C, але проводилися доопрацювання для сумісності з C++ компиляторами. В даний час Tesseract 3.0 збирається під Linux з GCC 2.95 і старше і під Windows з Visual C 2008 Express і старше (підтримка Visual C 6 була видалена у версії 3.0). На сьогоднішній день останньою версією є Tesseract 5.0, заснована на LSTM.

Sublime Text (Рис.2.1) - пропрієтарний текстовий редактор. Підтримує плагіни мовою програмування Python. Розробник дозволяє безкоштовно та без обмежень ознайомитися з продуктом, проте програма повідомляє про необхідність придбання ліцензії.

Sublime Text підтримує велику кількість мов програмування[9] і має можливість підсвічування синтаксису для C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, Rust, SQL, TCL та XML.

Крім тих мов програмування, які включені за замовчуванням, користувачі мають можливість завантажувати плагіни для підтримки інших мов.

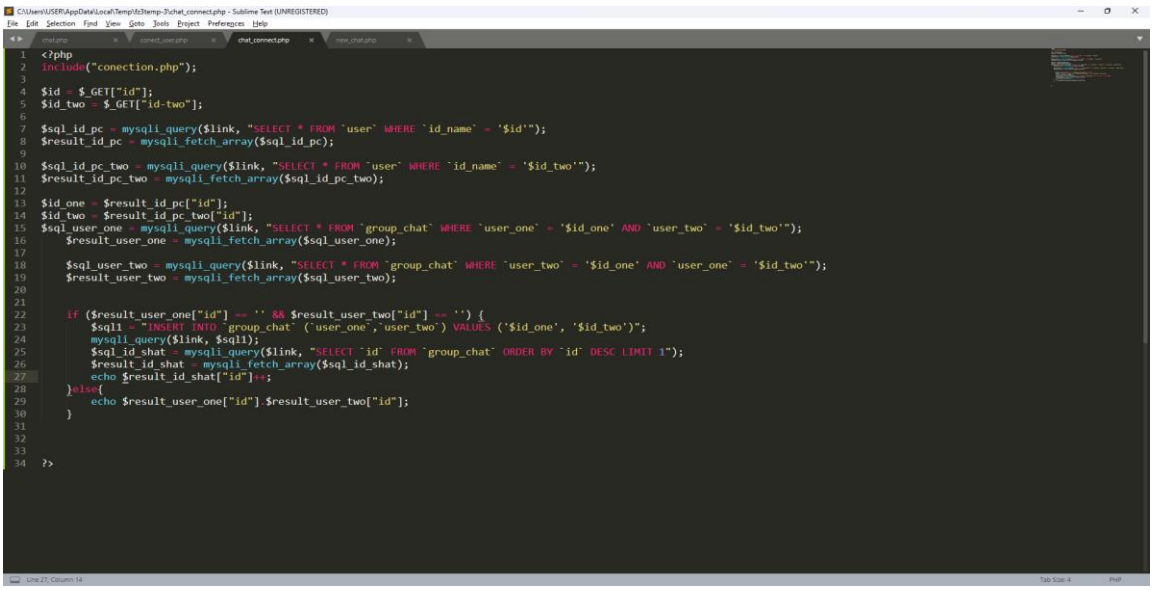


Рис.2.1 Sublime Text

Деякі можливості:

- Швидка навігація (Goto Anything)
- Командна палітра (Command Palette)
- API плагінів на Python
- Одночасне редагування (Split Editing)
- Високий ступінь настроюваності (Customize Anything)
- Індивідуальні налаштування проекту
- Широкі можливості налаштування за допомогою файлів налаштувань JSON, включаючи налаштування для конкретних проектів та платформ.
- Кросплатформенність (Windows, macOS і Linux) і допоміжні модулі для кросплатформенності, що підключаються.
- Сумісний з багатьма мовними граматиками з TextMate[8]
- Підтримка мов

Менеджер пакетів

Sublime Text може бути оснащений менеджером пакетів, який дозволяє користувачеві знаходити, встановлювати, оновлювати та видаляти пакети без

перезавантаження програми. Менеджер підтримує встановлені пакети у актуальному стані, завантажуючи нові версії з репозиторіїв. Крім того, він надає команди для активації та деактивації встановлених пакетів.

Інтерфейс

Редактор містить різні візуальні теми з можливістю завантаження додаткових.

Користувачі бачать весь свій код у правій частині екрана у вигляді міні-карти, при натисканні на яку можна здійснювати навігацію.

Існує кілька режимів екрана. Один із них включає від 1 до 4 панелей, за допомогою яких можна показувати до чотирьох файлів одночасно. Повноцінний (free modes) режим показує лише один файл без додаткових меню навколо нього.

Виділення стовпців та множинне виправлення

Виділення стовпців цілком або розстановка кількох покажчиків за текстом, що уможливлює миттєве виправлення. Покажчики поводяться, ніби кожен із них — один у тексті. Команди типу переміщення на знак, переміщення на рядок, вибірка тексту, переміщення на слово або його частини (CamelCase, поділений дефісом або підкресленням), переміщення на початок/кінець рядка і т. д., впливають на всі покажчики незалежно і відразу, дозволяючи редагувати складноструктурований текст швидко, без використання макрокоманд чи регулярних виразів.

Автодоповнення

Коли користувач набирає код, Sublime Text, залежно від мови, буде пропонувати різні варіанти для завершення запису. Редактор також автоматично завершує створені користувачем змінні.

Підсвічування синтаксису та висока контрастність

Темний фон Sublime Text призначений збільшення контрастності тексту. Основні елементи синтаксису виділені різними кольорами, які краще поєднуються з темним тлом, ніж зі світлим.

Підтримка систем збирання

Sublime Text дозволяє користувачеві збирати програми та запускати їх без необхідності перемикатися на командний рядок. Користувач також може налаштувати свою систему збирання та включити автоматичне збирання програми щоразу при збереженні коду.

Заготівлі (сніпети)

Збереження фрагментів коду, що часто використовується, ключові слова для їх запуску.

Перехід файлами

Навігаційний інструмент, який дозволяє користувачам переміщатися між файлами, а також усередині них за допомогою нечіткого пошуку.

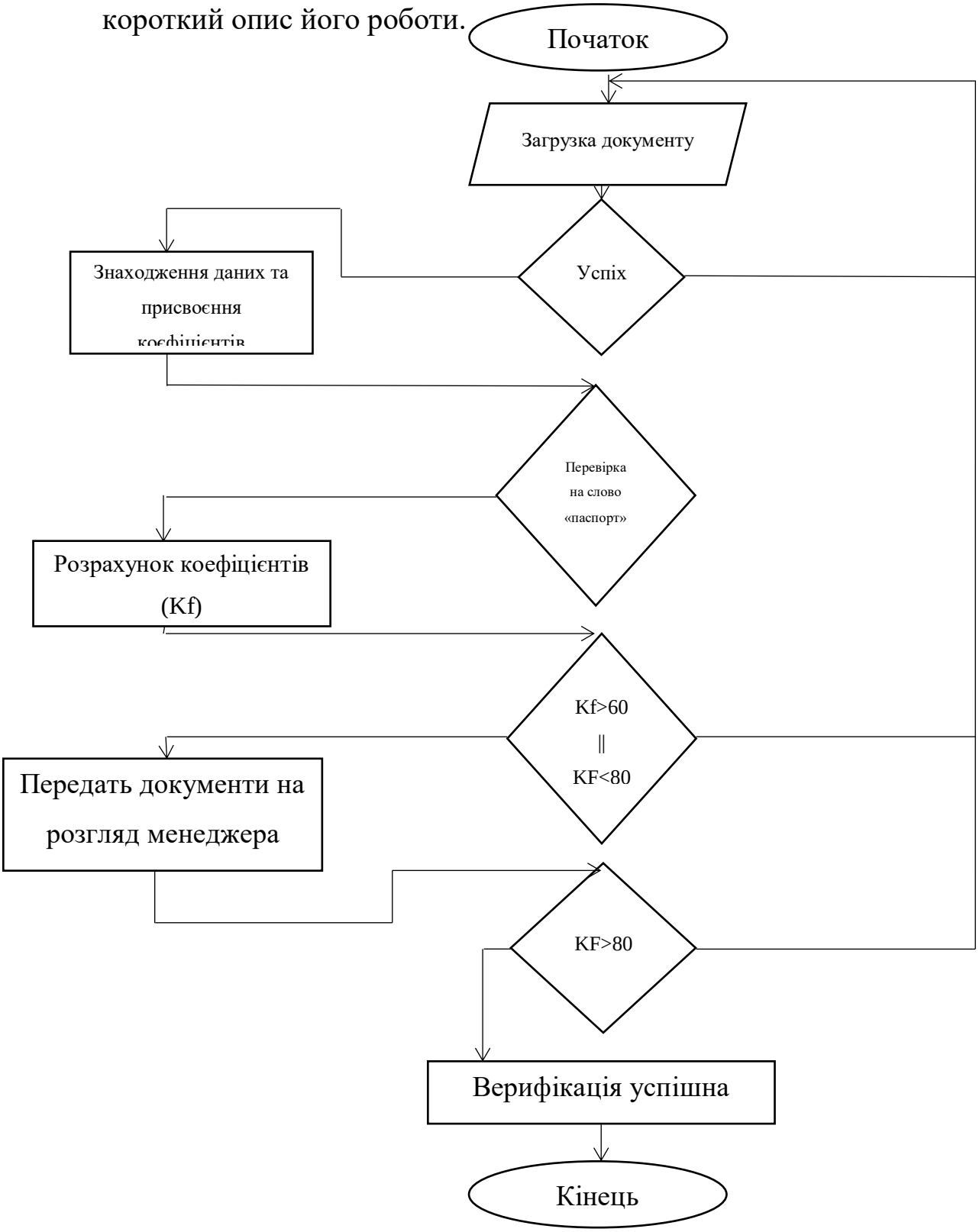
Інші особливості

- Додатково реалізовано функцію автозбереження, яка допомагає користувачам не втратити виконану роботу.
- Комбінації клавіш і інструмент навігації, що настроюються, дозволяють призначати свої комбінації клавіш для меню і панелей інструментів (тільки для першої версії, в другій і третій - Command Palette).
- Можливість пошуку в міру набору використовується для пошуку в документі.
- Функція перевірки синтаксису працює так само, перевіряючи коректність прямо під час введення.
- Є можливість автоматизації за допомогою макросів та повтору останніх дій.

Команди редагування, включаючи редагування відступів, переформатування параграфів та об'єднання рядків.

2.3 Побудова алгоритму роботи

В даному розділі буде показано алгоритм у вигляді блок схеми та короткий опис його роботи.



Алгоритм роботи:

1. Заповнення полів з особистою інформацією:
    a. ПІБ;
    b. Номер документу;
    c. Серія документу;
    d. Дата народження;
    e. Дата видачі;
    f. Орган видачі;
    g. ІНН;

2. Після вводу даних, потрібно загрузити фото документу. Зображення повинне бути або відскановане, або сфотографоване на камеру 8Мп, та з розширення 800x600, на даний момент такі камери мають навіть дешеві телефони. Характеристики вказані вище являються мінімальними, далі скрипт виділяє на зображенні текст:
    a. Паспорт (коеф. «обов'язковий»);
    b. Дата народження (коеф. «0.01»);
    c. ПІБ (коеф. «0.25»);
    d. Серія документу (коеф. «0.2»);
    e. Номер документу (коеф. «0.04»);
    f. Дата видачі (коеф. «0.1»);
    g. Орган видачі(коеф. «0.2»);
    h. ІНН(коеф. «0.2»).

2.4 Опис методу роспізнавання

Математична модель алгоритму реалізована наступним чином,

Маємо декілька показників тобто коефіцієнтів:

- Kf(p) – обов'язковий без якого алгоритм завершується
- Kf(pib) – ПІБ дані перевіряються співпадають чи вони з веденними (0.01)

- Kf(nd) – Номер документу дані перевіряються співпадають чи вони з веденними (0.01)
- Kf(sd) - Серія документу дані перевіряються співпадають чи вони з веденними (0.01)
- Kf(db) – Дата народження дані перевіряються співпадають чи вони з веденними (0.01)
- Kf(dv) – Дата видачі дані перевіряються співпадають чи вони з веденними (0.01)
- Kf(ov) – Орган видачі дані перевіряються співпадають чи вони з веденними (0.01)
- Kf(in) – ІНН дані перевіряються співпадають чи вони з веденними (0.01)
-

У випадку якщо Kf(p) існує виконується наступна формула:

Res=((Kf(pib)+Kf(nd)+Kf(sd)+Kf(db)+Kf(dv)+Kf(ov)+Kf(in))*100)/10

Res – результат відносності об’єктів на документі та підтвердження його.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Інтерфейс користувача

Для для верифікації потрібно натиснутии на блок для вибору файлу (рис.31.).

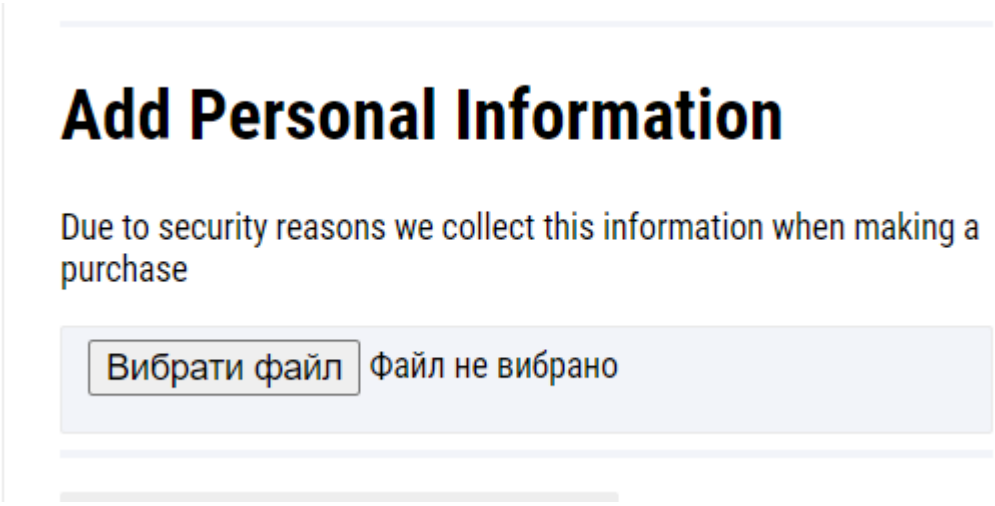


Рисунок 3.1 – Блок виклику вікна вибору

Після нього зявиться вікно в якому потрібно вибрати потрібний файл (рис.3.2).

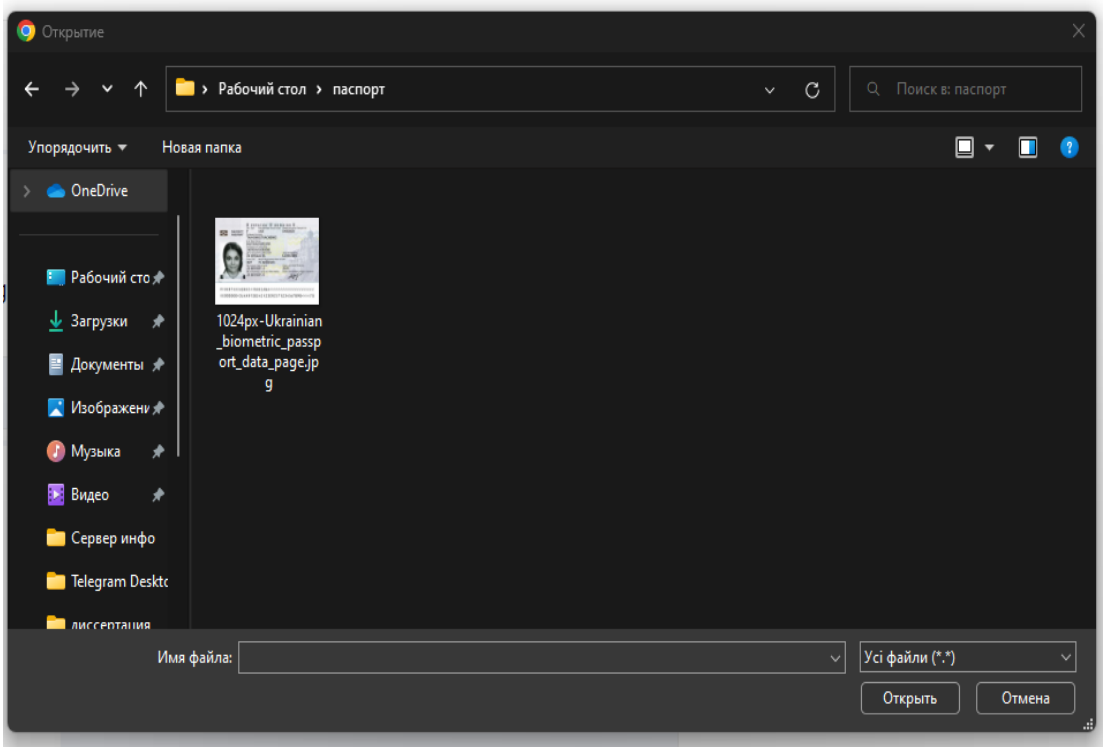


Рисунок 3.2 – Вибір файлу

Після вибору документ загружається на сервер та потребує підтвердження верифікаці(рис. 3.3).

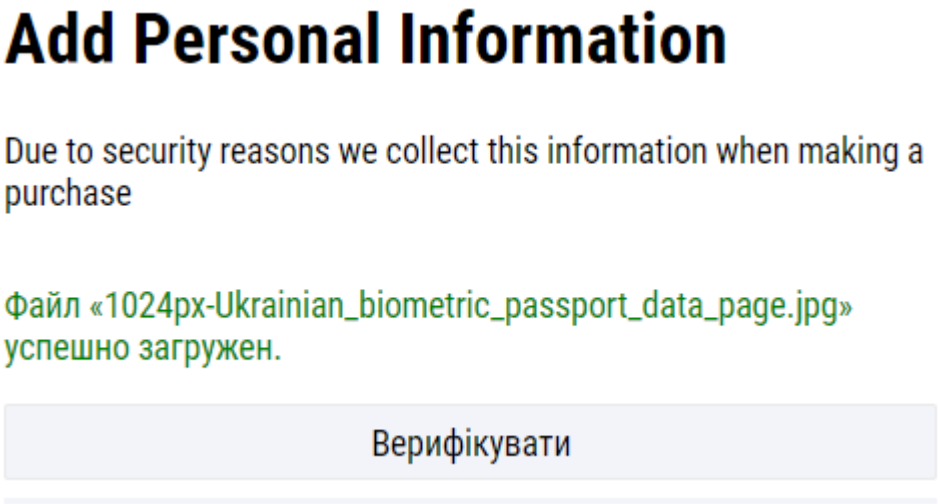


Рисунок 3.3 – Підтвердження верифікації

Після натискання на кнопку верифікаціє піде перевірка з відсотками (рис. 3.4) до завершеня її, та зявиться повідомлення про заключення легитивності документу(рис. 3.5., рис 3.6).

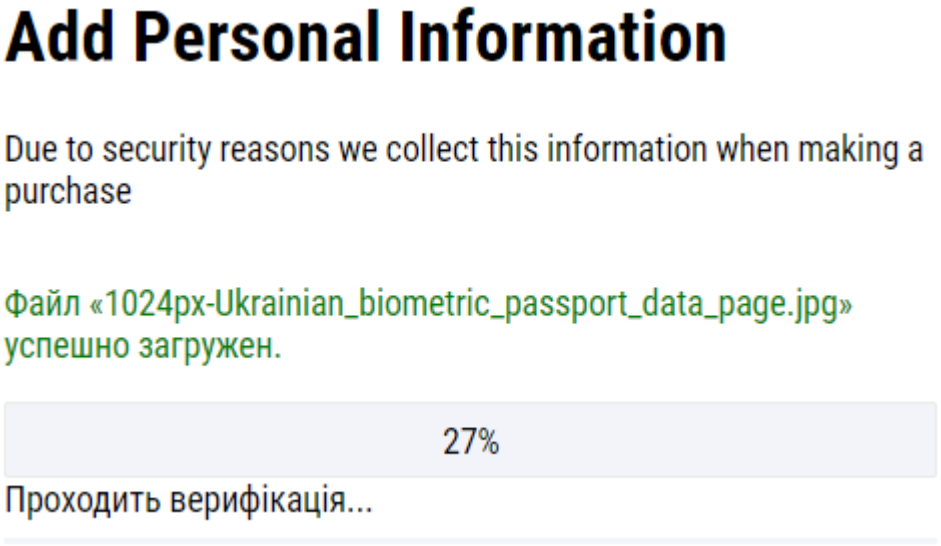


Рисунок 3.4 – Перевірка верифікації

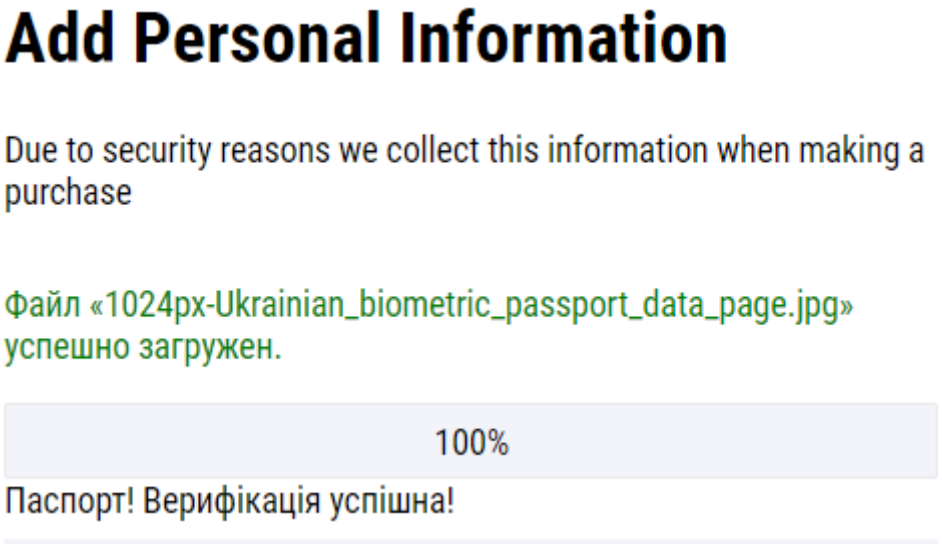


Рисунок 3.5 – Документ підтверджено

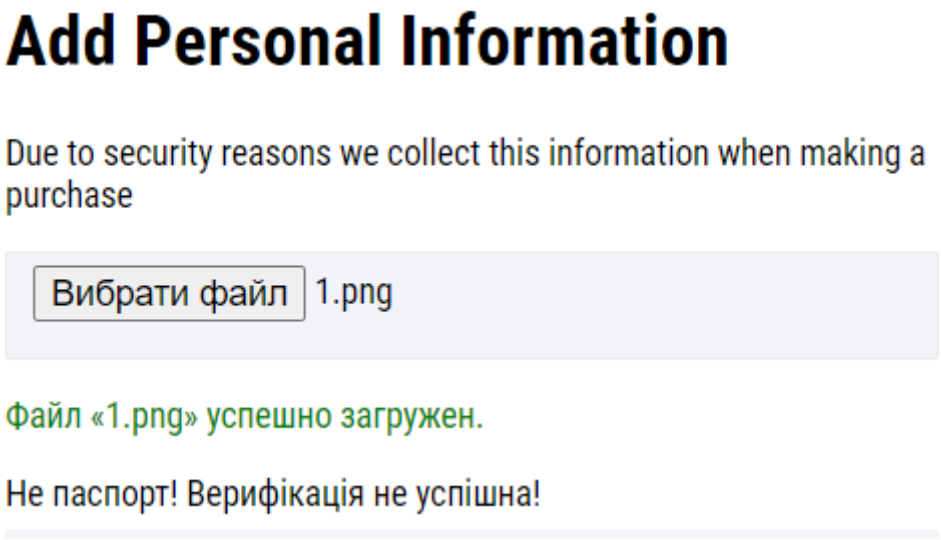


Рисунок 3.6 – Документ не підтвердженно.

3.2 Програмна реалізація функцій шифрування

Для реалізації чата було вибрано WEB версію верифікації документів, на повах JavaScript, та с використанням бібліотеки Tesaract :

Наступна функція є найголовнішою, вона відповадє за загрузку файла на сервер та сканування документу.

```
var formData = new FormData();
formData.append('file', $(".js-file")[0].files[0]);

$.ajax({
     type: "POST",
     url: 'upload.php',
     cache: false,
     contentType: false,
     processData: false,
```

```
                data: formData,
                dataType : 'json',
                success: function(msg){
                     if (msg.error == '') {
                          $(".js-file").hide();
                          $('#result').html(msg.success);
                            $("#ocr_button").fadeIn();

                             var            control            =
document.getElementById("ocr_url");
                               console.log(control);
                                 var files = control.files;

document.getElementById("ocr_button").addEventListener("click",
function() {

     $("#ocr_result").text('');

document.getElementById("ocr_result").innerHTML   =  "Проходить
верифікація...";

     Tesseract.recognize(

"uploads/"+files[0].name,
                                          'ukr',
                                          {   logger:   m   =>
$("#ocr_button").val((m['progress']*100).toFixed(0)+"%") }
                                       ).then(({ data: { text
} }) => {
                                             text             =
text.toLowerCase();
                                          console.log(text);
```

```
if(text.indexOf('пасп') > 0 || text.indexOf('пасл') > 0 ){

$("#ocr_result").text('Паспорт! Верифікація успішна!');
                                                    $("#step-sess-
3").removeAttr('disabled');
                                                    $("#step-sess-
3").on('click',function(){

     location.href='./test-ses.php';
                                                    })

                                                } else {

$("#ocr_result").text('Не паспорт! Верифікація не успішна!');
                                                    $(".js-
file").show();
$("#ocr_button").val('Верифікувати');

$("#ocr_button").fadeOut();
                                                }

//$("#ocr_result").text(text.toLowerCase());
                                                })

                                    });


                    } else {
                         $('#result').html(msg.error);
                    }

               }
          });
```

```
}
```

При загрузці методом Ajax , файл копіюється с пристрою клієнта на сервер, без пергрузагрузки сторінки, в цей час сразу же перевіряється формат документу, що заздалегідь блокує файли котрі можут нанести шкоду серверу та отримати файли інших клієнтів.

Після успішної загрузки файлу можно проводити верифікацію скануючи обєкт на потрібні словосполученя та символи для верифікації.

3.3 Підсумки реалізації проекту

Нині якісні системи розпізнавання тексту досить дорогі, тому досить поширені. Наприклад, FineReader 12 показує найкращий результат. Він розпізнає як скановані, так і сфотографовані зображення. Проте, мінімальна його вартість становить 80 євро за ліцензію на 10000 розпізнавань на рік. Ця система не має таких обмежень і може бути встановлена, наприклад, у бухгалтерії або відділі кадрів будь-якої компанії, для автоматизації збору та оновлення даних. Даний додаток на базі Tesseract дозволяє з досить високим ступенем точності розпізнавати символи, за наявності різних шумів і перешкод, завдяки розробленому алгоритму попередньої обробки зображень та системою обробки тексту, заснованої на словниковому контролі результатів. Безперечною перевагою системи є можливість швидкого доопрацювання під інший формат документів та інтуїтивно зрозумілий інтерфейс користувача.

В системні було реалізовано роспізнавання тексту на документі с подальшою перевіркою його на потрібні символо та словосполученя, для підтвердженя.

Також являється простота для встановлення як модуля API, дозволяє більш широкій аудиторії с меншим фінансовим обігом використовувати його.

ВИСНОВКИ

В результаті аналізу предметної області було детально розглянуто теоретичні відомості, що стосуються методів розпізнавання документів. Розглянуті існуючі методи роспізнавання та верифікації, взяті за увагу їх недоліки та сильні сторони, використвуючі дану інформацію було покращено метод.

В практичній частині роботи було успішно реалізовано алгоритм роспфзнавання. Вивчена інформація було обробленя та сприяла розробці API. Реалізавано у вигляді форми для загрузки файлів, використовуючи новий метод роспфзнаваня було значно покращено швидкість та точність підтвердження документів.

Було вирішено поставлені завдання:

1. Дослідити процеси та технології існуючих методів роспізнавання;
2. Проаналізувати існуючі системи які користуються найкращою репутацією в даній галузі;
3. Дослідити технології серверних та мережевих можливостей;
4. Застосувати розроблений модуль на практиці.

В результаті виконання поставлених задач, API для верифікації документів було виконано. Додаток готовий до встановлення в WEB системи та використання. А отже, поставлену мету роботи досягнуто.

ПЕРЕЛІК ПОСИЛАНЬ

1. Сінгх С. Книга шифрів. Таємна історія шифрів та їх розшифрування. М: Аст, Астрель, 2006. 447 с.
2. Matan O. Handwritten Character Recognition Using Neural Network Architectures. / O. Matan, R. Kiang, C. Stenard, et al. // Proceedings of the 4th US Postal Service Advanced Technology Conference. – 1990. – Vol. 9. – P. 1003-1011.
3. Mathematical Expression Recognition. [Электронный ресурс] URL: http://cat.prhlt.upv.es/mer/ (дата обращения: 20.02.2018).
4. Mouchere H. CROHME2011 : Competition on Recognition of Online Handwritten Mathematical Expressions. / H. Mouchere, C. Viard-Gaudin, D. H. Kim, et al. // 2011 International Conference on Document Analysis and Recognition. – 2011. – Vol. 4. – No. 1. – P. 1497–1500.
5. Mouchere H. ICFHR 2016 CROHME : Competition on Recognition of Online Handwritten Mathematical Expressions. / H. Mouchere, C. ViardGaudin, R. Zanibbi, et al. // 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). – 2016. – Vol. 5. – No. 1. – P. 607–612.
6. 30. Чочіа П.А. Сегментація зображень на основі аналізу відстаней у просторі ознак. // Автометрія. - 2014. - Vol. 50. - No. 6. - P. 97-110.

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Тема магістерської роботи:
« Розробка системи автоматизовоної валідації особистості людини для банкового ринку на основі нейронної мережі »

Виконала: Студентка групи ПДМ-62
Тертична Юлія Михайлівна

Керівник: к.т.н., доц., доцент, завідувач кафедри ІПЗ
Негоденко Олена Василівна

Київ - 2022

МЕТА, ОБ'ЄКТА ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ 2

Мета роботи: Підвищення ефективності перевірки особистості користувача в банківській системі валідації.

Об'єкт дослідження: Процес автоматизованої валідації на основі нейронної мережі.

Предмет дослідження: Технології штучних нейронних мереж для автоматизованої системи валідації.

АНАЛІЗ ІСНУЮЧИХ ІТ РІШЕНЬ ТА ЇХ МОДЕЛЕЙ

Модель	Додаток
Tesseract — вільна комп'ютерна програма для розпізнавання текстів, що розроблялася Hewlett-Packard з середини 1980х1900 років «пролежала на полиці».	Tesseract
Cloud Vision API ідея платформи – надати технології Computer Vision, у яких Google є безумовним лідером як сервіс.	Google Cloud Vision
FineReader PDF надає професіоналам можливість максимально підвищити ефективність на цифровому робочому місці.	ABBYY FineReader

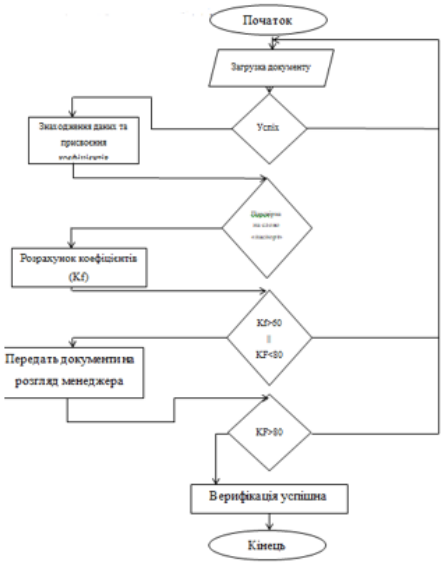
Необхідні коєфіцієнти та алгоритм

Алгоритм роботи:
Заповнення полів з особистою інформацією:

1. ПІБ;
2. Номер документу;
3. Серія документу;
4. Дата народження;
5. Дата видачі;
6. Орган видачі;
7. ІНН;

Після вводу даних, потрібно загрузити фото документу. Зображення повинне бути або відскановане, або сфотографоване на камеру 8Мп, та з розширення 800x600, на даний момент такі камери мають навіть дешеві телефони. Характеристики вказані вище являються мінімальними, далі скрипт виділяє на зображенні текст:

1. Паспорт (коеф. «обов'язковий»);
2. Дата народження (коеф. «0.01»);
3. ПІБ (коеф. «0.25»);
4. Серія документу (коеф. «0.2»);
5. Номер документу (коеф. «0.04»);
6. Дата видачі (коеф. «0.1»);
7. Орган видачі(коеф. «0.2»);
8. ІНН(коеф. «0.2»).



5

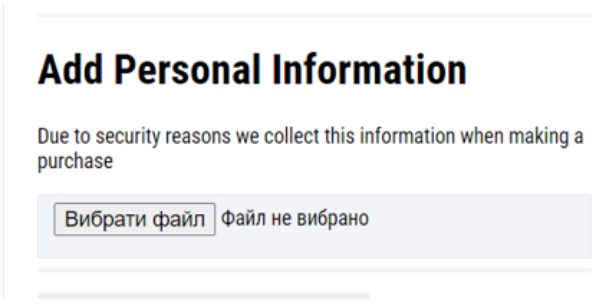
Реалізована математична модель алгоритму

Res=((Kf(pib)+Kf(nd)+Kf(sd)+Kf(db)+Kf(dv)+Kf(ov)+Kf(in))*100)/10

Kf(p) – обов’язковий без якого алгоритм завершується
Kf(pib) – ПІБ дані перевіряються співпадають чи вони з веденними (0.01)
Kf(nd) – Номер документу дані перевіряються співпадають чи вони з веденними (0.01)
Kf(sd) - Серія документу дані перевіряються співпадають чи вони з веденними (0.01)
Kf(db) – Дата народження дані перевіряються співпадають чи вони з веденними (0.01)
Kf(dv) – Дата видачі дані перевіряються співпадають чи вони з веденними (0.01)
Kf(ov) – Орган видачі дані перевіряються співпадають чи вони з веденними (0.01)
Kf(in) – ІНН дані перевіряються співпадають чи вони з веденними (0.01)

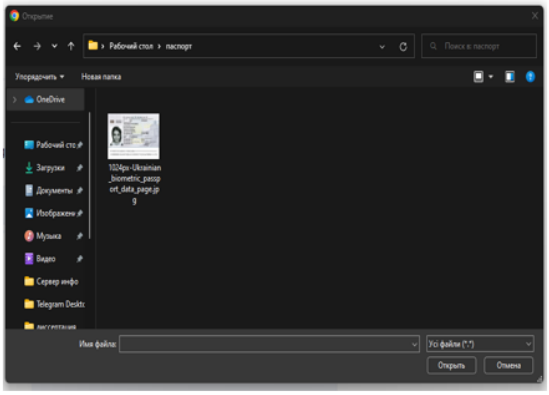
6

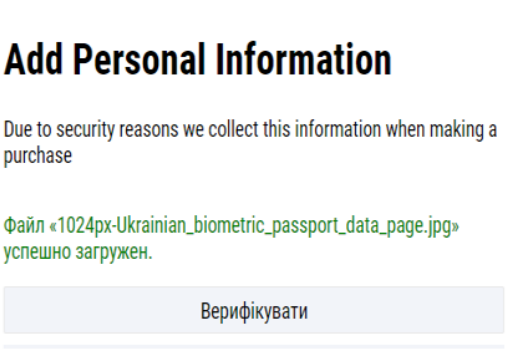
Інтерфейс користувача



Для для верифікації потрібно натиснути на блок для вибору файлу

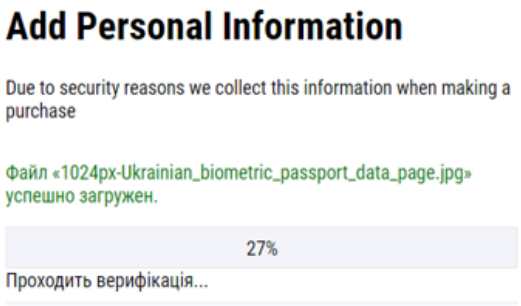
Інтерфейс користувача

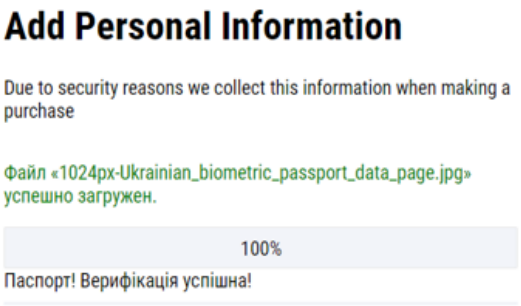




Після нього з’явиться вікно в якому потрібно вибрати потрібний файл. Після вибору документ загружається на сервер та потребує підтвердження верифікаці.

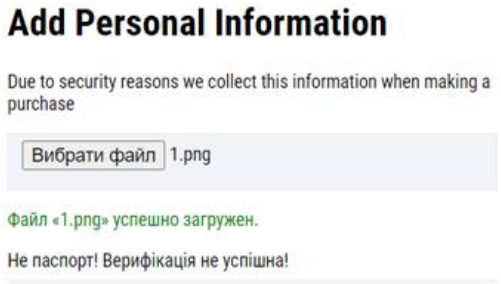
Інтерфейс користувача





Після натискання на кнопку, верифікаціє піде перевірка з відсотками до завершеня її, та з’явиться повідомлення про заключення легитивності документу

Інтерфейс користувача



Документ не являється паспортом

АПРОБАЦІЯ

Стаття:

Тертична Ю. М., Негоденко О. В., Дзядович О. С. Верифікація документів на веб ресурсі методом розпізнавання тексту.

ВИСНОВКИ

В практичній частині роботи було успішно реалізовано алгоритм роспфзнавання. Вивчена інформація було обробленя та сприяла розробці API. Реалізавано у вигляді форми для загрузки файлів, використовуючи новий метод роспфзнаваня було значно покращено швидкість та точність підтвердження документів.

Було вирішено поставлені завдання:

1. Дослідити процеси та технології існуючих методів роспізнавання;
2. Проаналізувати існуючі системи які користуються найкращою репутацією в даній галузі;
3. Дослідити технології серверних та мережевих можливостей;
4. Застосувати розроблений модуль на практиці.

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б

Програмний код

```
$(document).ready(function() {
function exp(){
     var  symbol_cur= $("select[name=ddl]").val();
     var symbol_cript = $("select[name=curs_ript]").val();
     $( "#exchangeRate-line" ).load( "curs.php?symbol_cript="+
symbol_cript+"&symbol_cur="+symbol_cur);
     var now_exp = $("#exchangeRate-line").text();
     var cashValue_val = $("input[name=cashValue]").val();
     var res_ecvip = cashValue_val / now_exp;
     $("input[name=btcValue]").val(res_ecvip.toFixed(8));
     $("#amountBTC").text(res_ecvip.toFixed(8));
    $(".proc_fee").text((res_ecvip.toFixed(8) *
(3/100)).toFixed(8));
    $(".block_fee").text((res_ecvip.toFixed(8) *
(0.5/100)).toFixed(8));
    $(".to_spend").text(((res_ecvip.toFixed(8) *
(3/100))+(res_ecvip.toFixed(8) * (0.5/100))).toFixed(8));
    $(".to_reciv").text(((res_ecvip.toFixed(8)*1) -
(res_ecvip.toFixed(8) * (3/100)) + (res_ecvip.toFixed(8) *
(0.5/100))).toFixed(8));
}
$("#start-sess").on('click', function(){
     var amount_curr = $("input[name=cashValue]").val();
     var amount_crypt = $("input[name=btcValue]").val();
     var email = $("#email").val();
     var btc = $("#btc").val();
     var exchangeRate = $( "#exchangeRate-line" ).text();
     var processing_fee =  $(".proc_fee").text();
     var blockchainFee = $(".block_fee").text();
```

```
      var totalSpend  = $(".to_spend").text();
      var totalReceive = $(".to_reciv").text();
      var form_value =
"amount_curr="+amount_curr+"&amount_crypt="+amount_crypt+"&email="+e
mail+"&btc="+btc+"&exchangeRate="+exchangeRate+"&processing_fee="+pr
ocessing_fee+"&blockchainFee="+blockchainFee+"&totalSpend="+totalSpe
nd+"&totalReceive="+totalReceive;
       $.ajax({
             type: "POST",
             data: form_value,
             url: "session-info.php", //Путь к обработчику
             success: function(html){
                         location.href='./personalDetails.php';
                   }
             })
})
$("#step-sess-2").on('click', function(){
      var firstName = $("input[name=firstName]").val();
      var lastName = $("input[name=lastName]").val();
      var date = $("input[name=date]").val();
      var nationality = $("input[name=nationality]").val();
      var phone = $("input[name=phone]").val();
      var  country_selector = $("#country_selector").val();
      var street = $("input[name=street]").val();
      var state = $("input[name=state]").val();
      var  city = $("input[name=city ]").val();
      var zip = $("input[name=zip]").val();
      var apt = $("#apt").val();
      var form_value =
"firstName="+firstName+"&lastName="+lastName+"&date="+date+"&nationa
lity="+nationality+"&phone="+phone+"&country_selector="+country_sele
ctor+"&street="+street+"&state="+state+"&city="+city+"&zip="+zip+"&a
pt="+apt;
```

```
      console.log(form_value);
       $.ajax({
            type: "POST",
            data: form_value,
            url: "session-info2.php", //Путь к обработчику
            success: function(html){
                        location.href='./documentVerif.php';
                    }
            })
})
$(".js-file").change(function(){
      if (window.FormData === undefined) {
           alert('В вашем браузере FormData не поддерживается')
      } else {
           var formData = new FormData();
           formData.append('file', $(".js-file")[0].files[0]);

           $.ajax({
                type: "POST",
                url: 'upload.php',
                cache: false,
                contentType: false,
                processData: false,
                data: formData,
                dataType : 'json',
                success: function(msg){
                     if (msg.error == '') {
                          $(".js-file").hide();
                          $('#result').html(msg.success);
```

```
                            $("#ocr_button").fadeIn();
                             var control =
document.getElementById("ocr_url");
                               console.log(control);
                                 var files = control.files;
document.getElementById("ocr_button").addEventListener("click",
function() {
                                          $("#ocr_result").text('');
document.getElementById("ocr_result").innerHTML  = "Проходить
верифікація...";
                                             Tesseract.recognize(
                                           "uploads/"+files[0].name,
                                           'ukr',
                                           { logger: m =>
$("#ocr_button").val((m['progress']*100).toFixed(0)+"%") }
                                         ).then(({ data: { text } })
=> {
                                             text =
text.toLowerCase();
                                           console.log(text);
                                           if(text.indexOf('пасп') >
0 || text.indexOf('пасл') > 0 ){

$("#ocr_result").text('Паспорт! Верифікація успішна!');
                                               $("#step-sess-
3").removeAttr('disabled');
                                               $("#step-sess-
3").on('click',function(){
               location.href='./test-ses.php';
                                               })
                                           } else {
                                           $("#ocr_result").text('Не
паспорт! Верифікація не успішна!');
```

```
                                                $(".js-file").show();
$("#ocr_button").val('Верифікувати');
                            $("#ocr_button").fadeOut();
                                                }
            //$("#ocr_result").text(text.toLowerCase());
                                            })
                                });
                    } else {
                         $('#result').html(msg.error);
                    }
                }
          });
     }
});
exp();
setInterval(exp, 1000);
})
<?php
session_start();

?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="./css/style.css">
    <link rel="stylesheet" href="build/css/intlTelInput.css">
    <link rel="stylesheet" href="build/css/countrySelect.css">
    <link rel="stylesheet" href="build/css/demo.css">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
```

```
    <title>Personal Deatails</title>
    <!-- <script src="tesseract/tesseract.js"></script> -->
</head>
<body>
<div class="main_wrapper">
    <div class="header">
        logo
    </div>
    <div class="order_wrapper">
        <div class="details_wrapper">
            <div class="order_status">
                <ul class="order_steps">
                    <li class="order_step">
                      <span>  Order
                        <br>
                        details
                        </span>
                    </li>
                    <li class="order_step">
                        <span> Personal <br>Details</span>
                    </li>
                    <li class="order_step">
                    <span>    Document <br>Verification
                        </span>
                    </li>
                    <li class="order_step">
                       <span> Place <br>the Order
                        </span>
                    </li>
                    <li class="order_step">
                       <span> Status <br>Check
                        </span>
                    </li>
```

```
                </ul>
            </div>
            <hr>
            <div class="payment_select">
                <h1>Add Personal Information</h1>
                <p>Due to security reasons we collect this
information when making a purchase</p>
                <div class="information_form">
                    <form action="#" class="form_personal_details">
                        <div class="name_surname_inputs">
                            <div class="name_input">
                                <!-- <label class="inputs_name"
for="firstName">Document</label>
                                <input type="file" id="js-file"
name="file"> -->

                                <input type="file" id="ocr_url"
class="js-file" />
                                 <div id="result" style="margin-
bottom: 0;">
                                    <!-- Результат из upload.php -->
                                </div>
                                <input type="button" id="ocr_button"
value="Верифікувати"  style="display: none;" />

                                <div id="ocr_result"></div>
                            </div>

                        </div>

                    </form>
                </div>
                <hr>
```

```
            </div>


            <div class="btns_list">
                <ul class="btns_list_holder">
                    <li class="btn_holder">
                        <button  class="agree_btn" id="step-sess-3"
disabled="disabled">CONTINUE TO PLACE THE ORDER</button>
                    </li>
                    <li class="btn_holder">
                        <button onclick="location.href='#'"
class="disagreeBtn">CANCEL</button>
                    </li>
                </ul>
            </div>
            inputs
        </div>
        <div class="cart">
            <div class="cart_info">
                <h2>Cart</h2>
                <div class="ssl_info">
                    <p class="sslSecure">SSL Secure</p>
                </div>
            </div>

            <div class="order_info">

                <p class="btc_amount">~<?php echo
$_SESSION['amount_crypt']; ?> for</p>
                <input id="cashInput" value="<?php echo
$_SESSION['amount_curr']; ?>" class="cash_amount" disabled/>
                <!--                <input type="text"
id="currencyInput">-->
```

```
</div>
<div class="summury_list">
    <h2>Summary</h2>
    <div class="exchangeRate">
        <p class="list-name">exchangeRate</p>
        <p><?php echo $_SESSION['exchangeRate']; ?></p>
    </div>
    <hr>
    <div class="fee  border-top">
        <p class="list-name">
            Processing fee:
        </p>
        <p>
            <?php echo $_SESSION['processing_fee']; ?>
        </p>

    </div>
    <hr>
    <div class="blockchainFee  border-top">
        <p class="list-name">
            Blockchain fee:
        </p>
        <p>
            <?php echo $_SESSION['blockchainFee']; ?>
        </p>
    </div>
    <hr>
    <div class="totalSpend  border-top">
        <p class="list-name">
            Total Spend:
        </p>
        <p>
```

```
                        <?php echo $_SESSION['totalSpend']; ?>
                    </p>
                </div>
                <hr>
                <div class="totalReceive  border-top">
                    <p class="list-name">
                        Total receive:
                    </p>
                    <p>
                        <?php echo $_SESSION['totalReceive']; ?>
                    </p>
                </div>
                <hr>
            </div>
        </div>
    </div>
    <footer class="footer_guest pt-4">
        <div class="container">
            <div class="footer_guest-license">
                <p class="small">
                    Global DLT Exchange S.R.O<br>
                    MSB Registration Number: 31000184534354
                </p>
            </div>
            <div class="footer_guest-payment_methods">
                <img src="./assets/img/logo-visa-colored.svg"
alt="visa-logo" width="56">
                <img src="./assets/img/logo-mastercard-colored.svg"
alt="mastercard-logo" width="56">
                <img class="pci-image" src="./assets/img/logo-
pci.svg" alt="pci-logo" width="56">
            </div>
            <div class="footer_guest-contact-schedule">
```

```
                <div class="footer_guest-contact">
                    <h4 class="font-primary text bold">United
Kingdom</h4>
                    <a href="tel:+44 9379992888" class="text">+44
9379992888</a>
                </div>
                <div class="footer_guest-schedule">
                    <p class="small">Weekdays: GMT: 06:00 -
20:00/EST: 01:00 - 15:00</p>
                    <p class="small">Weekends: Closed</p>
                    <p class="small">address</p>
                </div>
            </div>
            <div class="footer_guest-legalMenu">
                <div class="footer_guest-legal">
                    <div class="footer_guest-legal-item"><a
class="text" href="" target="_blank">AML Policy</a></div>
                    <div class="footer_guest-legal-item"><a
class="text" href="" target="_blank">Terms and
                        Conditions</a></div>
                    <div class="footer_guest-legal-item"><a
class="text" href="" target="_blank">Privacy Policy</a>
                    </div>
                    <div class="footer_guest-legal-item"><a
class="text" href="" target="_blank">Refund and <br
                            class="d-none d-lg-block"> Cancellation
Policy</a></div>
                </div>
                <div class="footer_guest-support">
                    <div class="footer_guest-support-item"><a
class="text"

href=""
```

```
target="_blank">Contact Us</a></div>
                </div>
            </div>
            <div class="footer_guest-logo">
                logo
                <img src=""
                     alt="logo">
            </div>
            <div class="footer_guest-disclaimer">
                <p class="small">
                    This website is operated by Global DLT Exchange
S.R.O (registration number 53 300 416), a company
                    regulated by the Financial Intelligence Unit
(“FIU”) in Slovak Republic and licensed to provide
                    virtual currency wallet services and virtual
currency exchange services (license number:
                    OU-BA-OZP1-2021/000409-5), with registered
address at Konventna 7, Bratislava, 81103, Slovak
                    Republic
                </p>
                <p class="small">
                    Note: purchasing or selling Cryptocurrency carry
significant risk. Prices can fluctuate on any given
                    day. Because of such fluctuations,
                    Cryptocurrency may gain or lose value at any
time. Cryptocurrency may be subject to large swings in
                    value and may even become absolutely worthless.
                </p>
            </div>
        </div>
    </footer>
</div>
```

```
<script>
    let cash =
document.getElementById("cashValue").addEventListener("input",
function () {
        document.getElementById("cashInput").value = this.value;
    });
    let currency =
document.getElementById("currencyValue").addEventListener("input",
function () {
        document.getElementById("currencyInput").value = this.value;
    });
</script>
<script src="build/js/intlTelInput.js"></script>
<script>
    var input = document.querySelector("#phone");
    window.intlTelInput(input, {
        // allowDropdown: false,
        // autoHideDialCode: false,
        // autoPlaceholder: "off",
        // dropdownContainer: document.body,
        // excludeCountries: ["us"],
        // formatOnDisplay: false,
        geoIpLookup: function (callback) {
            $.get("http://ipinfo.io", function () {
            }, "jsonp").always(function (resp) {
                let countryCode = (resp && resp.country) ?
resp.country : "";
                callback(countryCode);
            });
        },
        hiddenInput: "full_number",
        initialCountry: "auto",
        // localizedCountries: { 'de': 'Deutschland' },
```

```
        // nationalMode: false,
        // onlyCountries: ['us', 'gb', 'ch', 'ca', 'do'],
        placeholderNumberType: "MOBILE",
        // preferredCountries: ['cn', 'jp'],
        separateDialCode: true,
        utilsScript: "build/js/utils.js",
    });
</script>
<!--<script src="./js/country.js"></script>-->
<!--<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>-->
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.j
s"></script>
<script src="./build/js/countrySelect.js"></script>
<script src="./build/js/countrySelect.min.js"></script>
<script>
    $("#country_selector").countrySelect({
        defaultCountry: "jp",
        // onlyCountries: ['us', 'gb', 'ch', 'ca', 'do'],
        responsiveDropdown: true,
        // preferredCountries: ['ca', 'gb', 'us']
    });
</script>
<script
src='https://unpkg.com/tesseract.js@v2.1.0/dist/tesseract.min.js'></
script>
<script src="js/script.js"></script>
<script>



</script>
<!--
```

```
<script>
    window.onload = function() {
        document.getElementById("ocr_url").value = ""; // Сбрасываем
форму после перезагрузки
        var control = document.getElementById("ocr_url");
        control.addEventListener("change", function() {
            var files = control.files;
document.getElementById("ocr_button").addEventListener("click",
function() {
                document.getElementById("ocr_result").innerHTML  =
"Проходить розпізнавання…";
Tesseract.recognize(files[0].name).then(function(result) {
                    lang: "ukr"
                }).then(function(result) {
                    document.getElementById("ocr_result").innerHTML
= result.text;

                });

            });

        });

    };

</script> -->
</body>
</html>
```