**Плагіат** — привласнення авторства на чужий твір науки, літератури, мистецтва або на чуже відкриття, винахід чи раціоналізаторську пропозицію, а також використання у своїх працях чужого твору без посилання на автора (*Великий тлумачний словник сучасної української мови. — К.; Ірпінь; ВТФ «Перун», 2005. С. 977*).

Юридичне визначення цього терміна вперше з'явилося в українських нормативних документах лише в липні 2001 року в новій редакції Закону України «Про авторське право і суміжні права».

В освітньо-науковій сфері плагіат найчастіше виражається в публікації під власним іменем чужого тексту або у запозиченні фрагментів чужого тексту без зазначення джерела запозичення.

Загалом, плагіат можна поділити на **три основні типи**:

* **Копіювання** чужої наукової роботи, чи декількох робіт та оприлюднення результату під своїм іменем. Тобто, дослівне відтворення частини або всієї роботи, без дозволу автора і без вказівки джерела. Дослівне копіювання допустимо виключно у випадку цитування джерела (текст поміщається в лапки).
* **Створення суміші** власного та запозиченого тексту без належного цитування джерел. Включає в себе матеріали інших досліджень, процеси або таблиці. «Значущість» може бути охарактеризована кількістю і якістю скопійованого матеріалу. Якщо у вашій роботі використано ідеї чужої роботи, необхідно вказати на неї посилання.
* **Перефразування чужої праці без згадування оригінального автора**. Тобто, відтворення ідей інших авторів без дослівного копіювання, але без дозволу автора і без вказівки на джерело. Перефразування допустиме тільки в разі належного посилання на джерело та виключно зі збереженням оригінального змісту. Цей тип плагіату, якщо він зроблений більш-менш досконало на сьогодні важко виявити технічними засобами пошуку плагіату (програмами антиплагіату).
* **Повторне використання свого тексту.** Відтворення частин своєї роботи в дослідженні та публікацію статті як абсолютно нової роботи. Не дозволяється повторна подача статті в журнал, якщо рукопис вже подано на розгляд в інший журнал (або опубліковано в іншому журналі).

Плагіат можливий щодо:

* даних
* слів і фраз
* ідей і принципів

Масштаб плагіату може відрізнятися в залежності від того:

* Який обсяг чужої роботи був запозичений - кілька рядків, параграфів, сторінок, уся стаття?
* Що було запозичено / скопійовано - результати, методи, або розділ «Вступ»?

Також слід виділити такі види, як  фальсифікація (вигадування тих чи інших, наприклад, статистичних показників з подальшим вказуванням їх у якості власної роботи), реплікація (це процес копіювання даних з одного джерела на багато інших і навпаки, тобто своєрідне «тиражування» інформації без дозволу автора), а також републікація (повторне або багаторазове обнародування в іншому джерелі  чужої інформації за справжнім підписом автора й посиланням на джерело), рерайт (додавання до чужого матеріалу без дозволу автора  додаткової інформацію, з переробкою раніше обнародуваного матеріалу і заміною слів та  виразів) та компіляція (укладання з кількох чужих матеріалів свого, та редагування без дозволу – смислова, стилістична, граматична правка й скорочення чужого матеріалу).

При роботі над дослідженням слід пам'ятати про те, що вказівка ​​джерел інформації (включаючи роботи наукових керівників і ваші власні) є дуже важливою частиною процесу. Завжди вказуйте посилання на результати робіт, на підставі яких ви робите висновки в своїх дослідженнях.

**Правила цитування**

Цитування має використовуватися у всіх випадках, коли в роботі використовуються дані, взяті зі сторонніх джерел, а не отримані або створені безпосередньо автором. Порушення вказаних нижче правил і їх недотримання має розцінюватися як плагіат:

* якщо думка автора наводиться дослівно, то її слід взяти в лапки;
* якщо цитується великий уривок тексту, то він може не братися в лапки, натомість — виділяється або відбивається від решти тексту певним способом (набирається іншим кеглем, шрифтом, накресленням, відбивається від основного тексту більшими абзацними відступами тощо);
* допускається скорочення цитати, яке не веде до викривлення думки автора. Місце скорочення має бути відзначене в цитаті квадратними дужками з трикрапкою всередині;
* допускається перефразування цитати, зміна словоформ чи відмінків певних слів. В такому разі, цитата в лапки не береться, але в квадратних дужках обов'язково ставиться посилання на джерело (його порядковий номер зі списку використаної літератури, який додається до роботи);
* в списку використаної літератури завжди слід вказувати навіть ті джерела, які використовувалися під час підготовки роботи і вивчення теми, навіть якщо прямих посилань чи цитувань цих джерел в роботі нема.

**Академічна доброчесність** означає, що в процесі навчання чи досліджень, студенти, викладачі та науковці керуються, передусім, принципами чесності, чесної праці та навчання. Плагіат, списування, несанкціоноване використання чужих напрацювань є неприйнятним і жодним чином не толеруються в спільноті.

**Оригінальність тексту** — це поняття, протилежне плагіату. Чим більше плагіату в тексті, тим менше його оригінальність і навпаки - чим менше плагіату, тим унікальність даного тексту вища. Справедливою є наступна формула:

**% плагіату + % оригінальності = 100%**

Наприклад, якщо програма перевірки рівня плагіату (антиплагіат) показує, що оригінальність певного тексту становить 70%, це означає, що 30% цього тексту - неоригінальні. Тобто з точки зору програми перевірки 30% у цьому випадку - плагіат.

Перевірити текст на унікальність можна кількома способами:

* за допомогою спеціальних програм;
* за допомогою сайтів, які надають такі послуги.

**Для чого потрібна унікальність?**

Користувачів інтернету цікавить нові, цікаві та незвичайні дані. Тексти на сайтах повинні бути завжди унікальними, рівних яким пошуковики не зможуть знайти в інтернеті. Копіювання текстів з інших ресурсів буде вважатися плагіатом. Пошукові системи блокують сайти з плагіатом, якщо там немає посилань на першоджерело. У статті не повинно бути багато посилань. Інакше пошуковики візьмуть статтю за спам і будуть її блокувати.

Пошукові машини не можуть зберігати всю інформацію. У них не так багато місця на жорстких дисках. Пошуковики видаляють безліч копій, а залишають тільки оригінали. Якщо в процесі індексації зустрінеться не унікальний контент, то вони його просто ігнорують. В пошукових машинах на жорстких дисках зберігається тільки унікальна інформація.

**Антиплагіат** — комп’ютерна програма, яка виявляє збіг деяких частин тексту. Самі по собі тексти можуть не бути плагіатом. Це можуть бути посилання, цитати і викладення і т. д. При написанні статті, автори можуть використовувати власний текст і різні джерела інформації. Багато авторів пишуть тексти на замовлення, і вони відповідальні за виконувану роботу. Неважливо, буде це стаття про модний бренд або стаття про будівництво, будь-який виконавець буде зацікавлений в унікальності своєї статті. Жоден клієнт не захоче мати на своєму сайті тексти, на які можуть пред’явити права конкуренти.

При купівлі або продажу своєї статті жодна людина не в змозі знайти будь-які копії в інтернеті з-за великої кількості інформації. Та й часу на це у нього не вистачить. Ось тут і знадобиться програма-антиплагіат, яка зможе чітко проаналізувати інформацію, перевірити статтю на унікальність. Така перевірка допоможе у статті уникнути копіювання фраз або мовних зворотів або просто повторень.

Алгоритм антиплагіат визначає ступінь унікальності документів. База даних програми — це відкриті джерела інтернету, включаючи базу рефератів. Перевірка тексту починається з завантаження статті. Програма сприймає досліджуваний текст, як набір пропозицій. Кожне речення складається з набору слів. Під час перевірки програма порівнює кожне речення в тексті з пропозиціями, які знаходяться в базі даних системи. Якщо слова або їх порядок в реченні співпадуть зі словами в реченнях з бази даних, то це буде вважатися плагіатом.

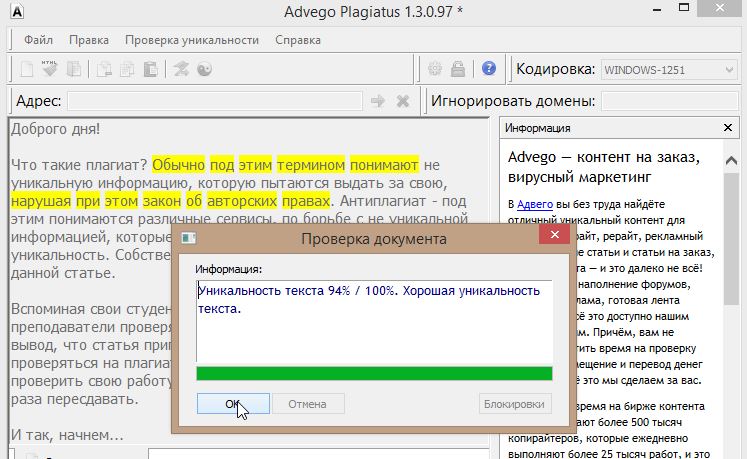
Якщо програма антиплагіат не знайшла відповідників в цих пошукових системах, то текст, що перевіряється, є унікальним. Якщо текст повністю або по шматках скачаний з інтернету, то програма вкаже посилання на всі першоджерела і відсоток унікальності тексту.

**Програми для перевірки тексту на унікальність** (безкоштовні):

1) **Advego Plagiatus (**http://advego.ru/plagiatus/)

Одна з кращих і найшвидших програм для перевірки будь-яких текстів на унікальність. Для перевірки тексту, досить просто його скопіювати у вікно з програмою і натиснути кнопку перевірки, сайти, де знайшлися такі ж шматки тексту, відображаються в нижньому вікні програми.

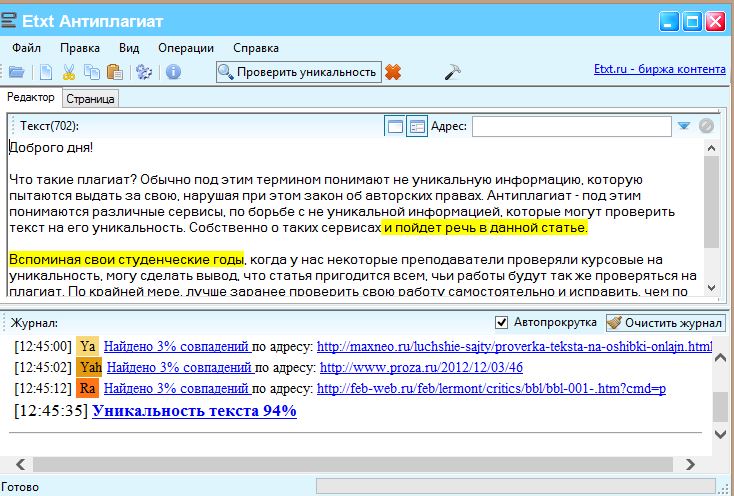
**Мінуси:** для роботипотрібна авторизація і установка на ПК, максимальна кількість символів – 100 000 символів, що не підходить для перевірки великих робіт(дипломів, курсових, рефератів більших за обсягом).



2) **Etxt Antiplagiat (**http://www.etxt.ru/antiplagiat/)

Аналог Advego Plagiatus, правда, перевірка тексту триває довше і перевіряється він більш ретельніше. Зазвичай, у цій програмі відсоток унікальності тексту нижче, ніж у багатьох інших сервісах. Користуватися їй так само просто: спочатку потрібно скопіювати текст у вікно, потім натиснути кнопку перевірки. Через десяток-другий секунд програма видасть результат.

**Мінуси:** потрібна авторизація і установка на ПК, текст не повинен перевищувати більше 5 000 символів, що підходить краще для превірки зовсім невеликих робіт. Після використання ліміту на день перевірка буде коштувати 50 коп. за кожні наступні 1 000 символів. До того ж, зі збільшенням символів, час перевірки зростає у геометричній прогресії.

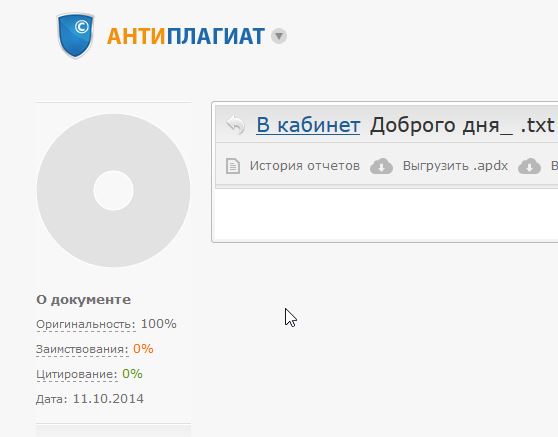


**Онлайн сервіси перевірки на плагіат.**

Подібних сервісів десятки, якщо не сотні. Всі вони працюють з різними параметрами перевірки, з різними можливостями та умовами. Найцікавіші сервіси, якими користується більшість:

1) **ANTIPLAGIAT** (http://www.antiplagiat.ru/) — ідея сервісу полягала спочатку в перевірці на плагіат курсових та дипломних робіт. Перевіряти можна як простий текстовий файл (завантаживши його на сайт), так і просто скопіювавши текст у вікно. Перевірка проходить досить швидко. До кожного тексту, який ви завантажили на сайт буде наданий звіт. Він дійсно корисний для студентів і аспірантів, але дуже слабенький при перевірці звичайних текстів. У загальному і цілому, можете користуватися для будь-якої початкової перевірки на унікальність, але до результатів варто ставитися з підозрою. Краще підкріпити їх результатами в інших системах.

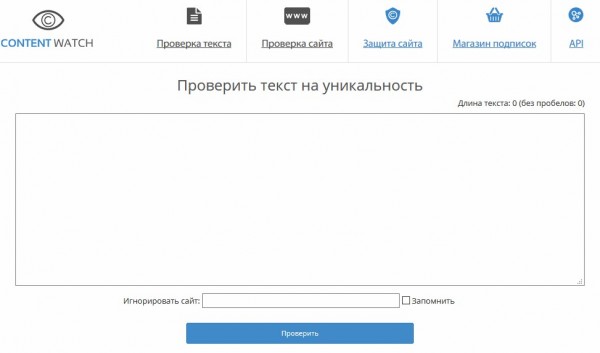
**Мінуси:** слабкий для перевірки звичайних текстів, до результатів варто ставитися з підозрою і краще підкріплювати їх результатами в інших системах. Без реєстрації можна перевіряти за раз до 5 000 символів, що суперечить самій ідеї даного сервісу.



2) **CONTENT-WATCH (**http://www.content-watch.ru/text/) — ще один онлайн сервіс. При перевірці на унікальність він використовує власний алгоритм пошуку в Інтернеті сайти можуть містити матеріали повні або часткові копії заданого тексту.

Реєструватися для перевірки на сайті не потрібно. На основі запропонованих варіантів підраховується загальна унікальність тексту у відсотках, а також унікальність щодо кожної знайденої сторінки з збігами. Є можливість подивитися, які частини тексту були знайдені на кожній з проаналізованих сторінок. При цьому існує відмінна можливість ігнорувати певний сайт.

**Мінуси:** до обмежень відноситься довжина тексту до 3 000 символів (розширюється до 10 000 символів після реєстрації); до 5 запитів в день на одного користувача (7 після реєстрації). При підключення тарифу в 70 грн./міс. Використання зростає до 20 000 символів і 15 запитів від одного користувача.

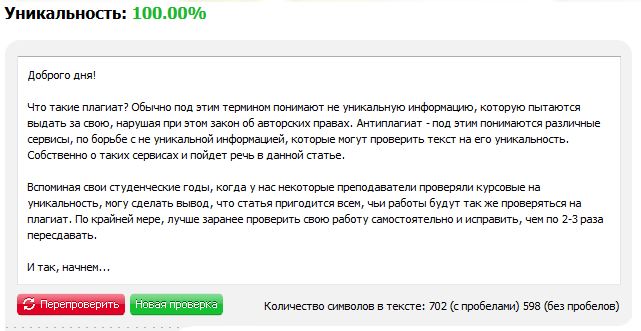


3) **TEXT.RU (**http://text.ru)— за заявами розробників, це найбільш просунутий сервіс. Він здійснює перевірку на плагіат онлайн за алгоритмом, який вміє наступне:

* захищати від неякісного рерайту зі зміною кожного п'ятого чи четвертого слова (в текстах, написаних так само, буде виявлений високий відсоток збігів з першоджерелом);
* також він виявляє просту перестановку слів, фраз і пропозицій місцями; не допоможе і зміна відмінків, часів і інших граматичних категорій слова; не зіграє ролі додавання в вихідне пропозицію нових слів.

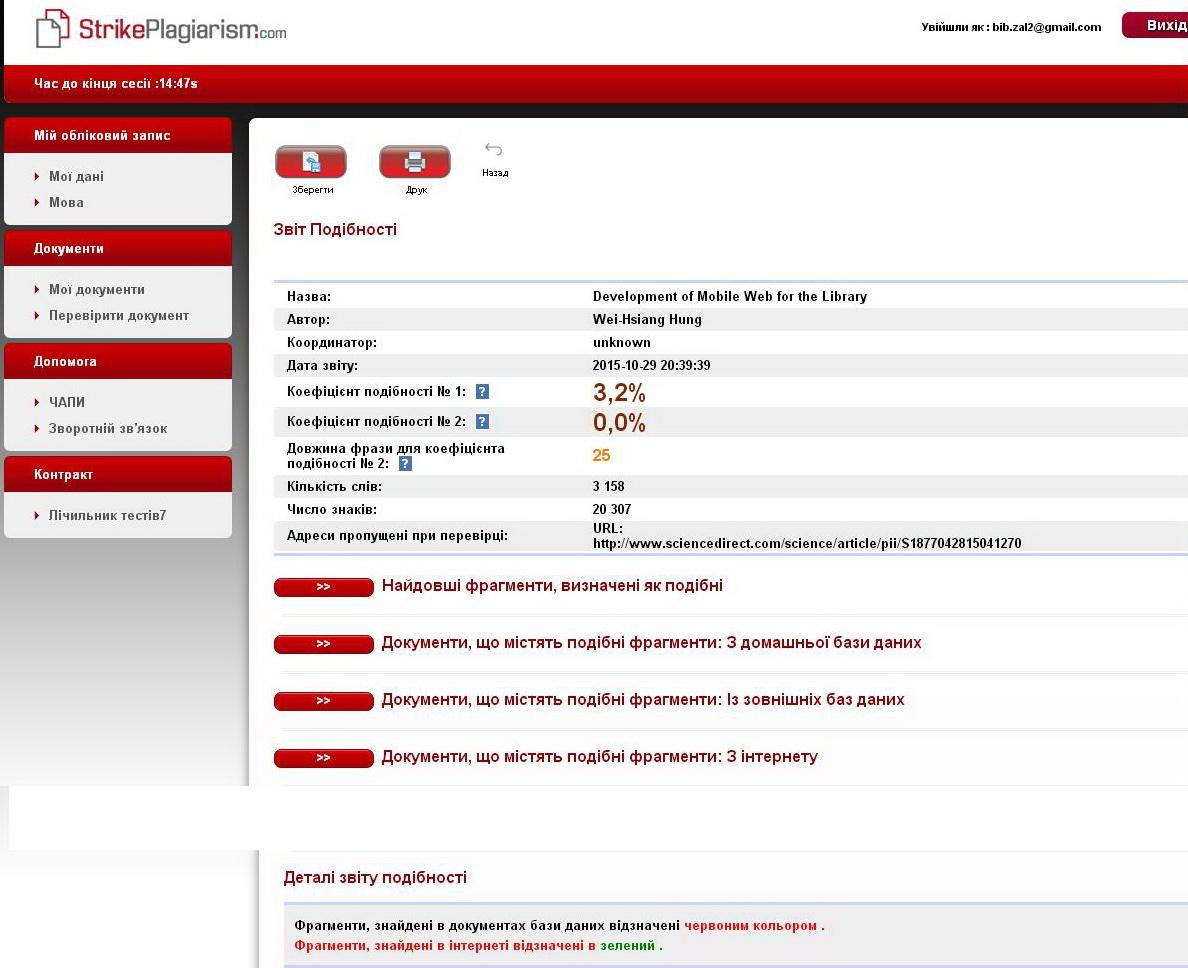
Виходить, що це найкраще з того, що можуть запропонувати онлайн сервіси. Безкоштовна онлайн перевірка тексту, реєструватися не потрібно.

**Мінуси:** безкоштовнодо 50 000 тис. символів в день, що є досить хорошим результатом для онлайн сервісу, але всерівно цього не вистачає для швидкої перевірки багатьох робіт. До того ж, щоб використовувати API даного продукту ти повинні купити PRO-акаунт.



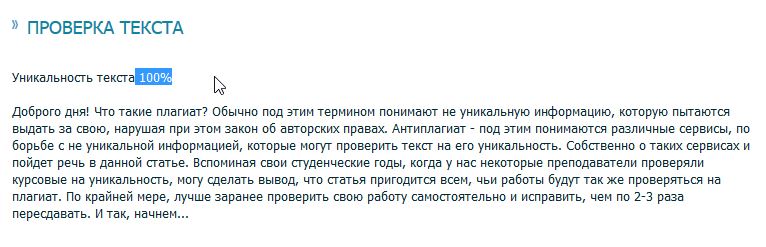
4) **StrikePlagiarism.com** ― антиплагіатна інтернет-система, яка автоматично перевіряє оригінальність тексту. У системи простий і зручний для користування інтерфейс. Документи можна завантажувати в систему в багатьох популярних форматах (DOC, ODT, TXT, PDF), відсутні обмеження на обсяг документу. Текст порівнюється з ресурсами інтернету та базою даних системи.

**Мінуси:** користування без реєстрації неможливе, безкоштовне користування тільки перший місяць.



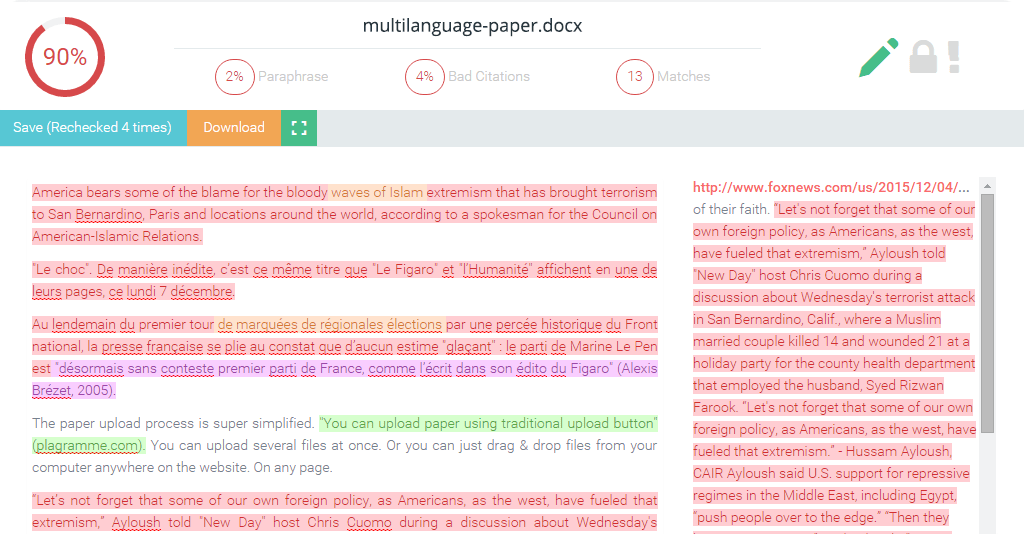
5) **ANTIPLAGIAT.SU** (http://antiplagiat.su**/)** Дуже проста і швидка перевірка. Реєструватися не потрібно, але і чекати від сервісу понад інформативності теж не потрібно. Після перевірки, з'являється просте вікно в якому показаний ваш текст і його унікальність. До речі, перевіряти можна і текст, і текстові файли.

**Мінуси:** сумнівний результат перевірки і часто не співпадає з більш відомими сервісами.



6) **PlagiarismCheck** розроблена в 2011 році і з того часу обслуговувала понад 77 000 користувачів з 72 країн світу.

**Мінуси:** безкоштовно тільки перша сторінка, надалі вам потрібно буде зареєструватись і сплачувати 180 грн за кожні 20 сторінок тексту.



Отож, можна зробити висновок, що сучасні сервіси перевірки антиплагіату не дають можливості користуватись ними у такому обсязі в якому нам потрібно. До того ж, дивлячись на ситуацію в інтернет мережі, було б нерозумно використовувати програму під російським доменом, оскільки таких сайтів більшість.

Саме тому було прийняте рішення реалізувати свою систему яка буде працювати на базі існуючих рішень, матиме швидкий час перевірки і буде абсолютно безкоштовною.

Шукаючи метод реалізації даного завдання знайшов декілька математичних варіантів які використовуються у сучасних системах перевірки антиплагіату і які можна використати у своїй програмі:

**1. Метод Шинглів**

**2. Метод векторного алгоритму пошуку.**

**3. Метод бінарного алгоритму пошуку.**

Розглянемо детальніше перший метод.

**Алгоритм шинглів** (від англ. Shingles - лусочки) - алгоритм, розроблений для пошуку копій і дублікатів розглянутого тексту в веб-документі. Інструмент для виявлення плагіату. Уді Манбер в 1994 р першим в світі висловив ідею пошуку дублікатів, а в 1997 р Андрій Бродер, високопоставлений співробітник Yahoo! запропонував використовувати її у галузі лінгвістичного аналізу тексту. Саме він оптимізував і довів алгоритм до логічного завершення, давши ім'я даній системі - «алгоритм шинглів».

**Шингл** – це частинка, цеглинка – одне з багатьох значень англійського слова shingle. У сфері просування сайтів шингл означає частинки-цеглинки, з яких будується текст, і є основою для самого надійного методу перевірки унікальності тексту. Шингл в цьому значенні має пряме відношення до лінгвістичного аналізу тексту. Користуючись дослідженням тексту за допомогою шинглів можна безпомилково відокремити унікальний текст від синонімізованого контенту. В даний час, коли SEO-оптимізація набуває професійного рівня, питання шинглів і роботи з ними стає ще більш актуальнішим.

**Підготовка тексту**

**Шингл** – це невеликий фрагмент тексту, оброблений за спеціальною методикою для аналізу. Ця методика – канонізація – полягає в тому, що текст cпрощується на всі несуттєві для розуміння і граматики слова (сполучники, прийменники, вигуки) і розділові знаки.

**Cкладання шинглів**

Другий етап роботи з текстом: безпосереднє виділення шинглів. Для цього в першу чергу потрібно визначити довжину шингла. Чим менший шингл, тим більше роботи і тим точніший аналіз. Мінімальний шингл дорівнює трьом словами, максимальний – восьми. Вибирати довший шингл не має сенсу, тому що, така перевірка буде мати велику похибку. Одне з правил складання шингла – метод накладання, тобто з захопленням як мінімум одного слова з попереднього шингла. Саме це дасть можливість точної та якісної перевірки всіх слів.

**Алгоритм шинглів**

Далі програмістом розраховується контрольна сума тексту. Алгоритм шингла передбачає порівняння шинглів двох різних текстів, визначаючи, чи не є один з них, дублікатом іншого. Пошукові системи, в які закладено програмний метод шинглової перевірки, обдурити дуже важко.

Вони знаходять подібні документи і порівнюють число збігів шинглів, чітко виявляючи відсутність унікальності. Тому при розмноженні тексту потрібна “хірургічна” робота по сінонімізації, розбавленню і структуруванню тексту, заміні фразеологічних висловів, так щоб зробить його не схожим на оригінал та надати унікальності.

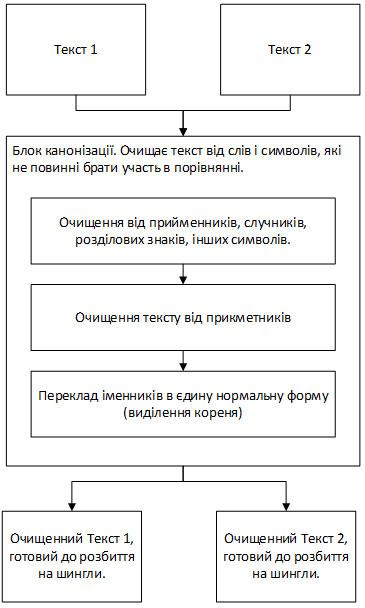


Рис1. Алгоритм Шинглів

Таким чином, розмножуючи контент, необхідно ретельно працювати над розведенням, сінонімізаціі і структуруванням майбутнього текстового вмісту для готової сторінки. Мета оптимізатора на даному етапі - зробити все можливе для того, щоб вихідний текст не був схожий на вже наявний в мережі. Ризик потрапити під фільтр пошукової системи через неунікального матеріалу пропадає лише тоді, коли використовуються виключно унікальні статті. Низький показник унікальності є головною причиною зниження позицій в рейтингу видачі. Тому просування неунікальні статтями не має ніякого сенсу.

**Приклад використання алгоритму методу шинглів**

Розглянемо як приклад дві злегка змінені витяги з вірша А.С. Пушкіна  
Оригінальний текст:

"Буря мглою небо кроет,  
Вихри снежные кружа,  
То как зверь она завоет,  
То заплачет как дитя"

Трохи виправлений текст  
"Буря белым землю кроет,  
Вихри снежные кружа,  
То как лев она завоет,  
То заплачет как дитя"

В якості кроку виберемо слово. Довжину підстроки візьмемо рівну 5 словами. Складати рядки будемо в стик (один за одним). В результаті отримаємо кодований текст довжиною в 5 чисел.

  
Рис2. Приклад компонування тексту методом шинглів

Ось у нас вийшов набір слів для першого випадку:  
БурямглоюнебокроетВихри | снежныекружаТокакзверь | оназавоетТозаплачеткак | дитя

хеш:  
a7bdbcb13968a694f626a5682b7f2dfd | 0e5aa06baba90d7c851f9a0450a60222 | c0c522529b0e810f73b210cc972e9966 | 95ed3beeb9bc9ff61affd4421a24c44f | 9c793e2986f7ee89f93953e3fbcab408

і другого:  
БурябелымземлюкроетВихри | снежныекружаТокаклев | оназавоетТозаплачеткак | дитя

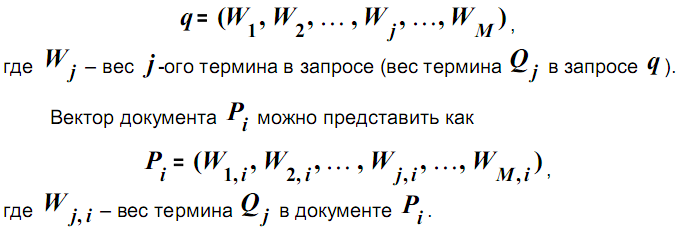
хеш:  
de5790caa3ee48c73f62e49000121c6f | 11da4405827ce2d70015f98a10563e1c | c0c522529b0e810f73b210cc972e9966 | 7172b4096aa49236a2f7edd298a47de2 | 690e13e46c9738d430d90570888d428f

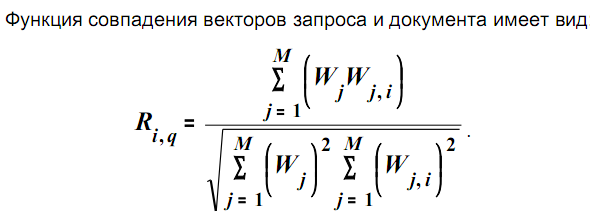
В результаті, у нас вийшов один збіг - третє число (c0c522529b0e810f73b210cc972e9966). Це збіг показує те, що між двома текстами схожість становить не менше 25%. Звичайно для такого маленького тексту, можна було зменшити крок, але і при таких початкових параметрах це є хорошим прикладом.

**Метод векторного алгоритму пошуку.**

Векторний алгоритм пошуку, званий IDF TF × -алгоритми, є одним з найпоширеніших. Він заснований на векторній моделі інформаційного масиву, в якій для визначення міри близькості документів і запитів використовується значення косинуса кута між їх векторами в багатовимірному просторі інформаційного масиву.

Запити та документи в векторної моделі представляють множинами наборів зважених термінів. Вектор запиту q в такому випадку буде виглядати так:





Під час пошуку часто буває важливо отримати по можливості найбільше значення повноти, тобто видати максимальну частину релевантних документів, наявних в масиві. Вичерпний пошук може знадобитися, наприклад, експертам організації, яка реєструє винаходи, яким необхідно скласти список всіх існуючих патентів. Збільшення числа релевантних документів зазвичай призводить до видачі додаткових нерелевантних документів, тобто знижується його точність .

Для поліпшення повноти пошуку необхідні додаткові збіги термінів запиту і документа. Це досягається використанням додаткових термінів-заступників. Терміни-заступники або додаються до вже існуючих термінів запитів і документів, або використовуються замість них.

Найбільш відомим методом тут є застосування словника синонімів (тезаурус), в якому терміни згруповані в класі синонімії (класи еквівалентності).

За допомогою тезауруса можна замінити кожен наявний в початковий момент пошуку термін ідентифікаторами відповідних класів тезауруса. При використанні іншого підходу ідентифікатори цих класів можна додавати до вихідних термінів. У будь-якому випадку мета полягає в тому, щоб отримати додаткові збіги для тих термінів запиту і документа, які віднесені до одних і тих же класах тезауруса. Самі ці терміни можуть бути і різними.

**Метод бінарного алгоритму пошуку.**

Методи пошуку в відсортованих таблицях покажчиків засновані на алгоритмі бінарного пошуку. Також, вони використовуються для знаходження відсотку унікальності тексту. Розглянемо принцип роботи детальніше:

Нехай впорядкований масив *х* з *п* елементів містить значення:

5 7 11 18 26 32 44 57 81 90 94 97 107 116 129 147 179 і нехай заданий аргумент пошуку - *ключ,* рівний, наприклад, 129.

Ідея алгоритму бінарного пошуку така:

• порівняти аргумент пошуку ключ зі значенням середнього елемента *х [середовищ]*масиву *х,* де середовище = *[П / 2],* а [з] ціла частина числа с.

• якщо вони рівні, то пошук завершено, інакше, якщо ключ < *х [середовищ],* виконати аналогічним чином пошук в позиціях масиву *х,* попередніх позиції середовищ, в іншому випадку, якщо ключ> *х [середовищ],* виконати аналогічним чином пошук в позиціях масиву *х* наступних за позицією середовищ. Виключити з подальшого розгляду частину масиву дозволяє той факт, що масив впорядкований.

Проілюструємо процес бінарного пошуку. Число елементів масиву *п =* 17, тоді*[П / 2]* = 8. Тому спочатку виконується порівняння *середовищ* з *x*8 = 57. Так як *середовищ > x8,* то зона пошуку на наступному кроці обмежується ділянкою від 9-го елемента до 17-го. Ця ділянка складається з дев’яти елементів і його серединою є елемент *х13* = 107 ([(9 + 17) / 2] = 13) Оскільки *середовищ> x*13, то зона пошуку обмежується ділянкою від 14-го до 17-го елементу. Його серединою є елемент *x15.* На цьому процес пошуку завершений, так як *х15 = середовищ.* Відобразимо процес пошуку елемента, *ключ* = 129, виділяючи у вигляді підкреслення на кожному кроці зони пошуку:

**Ітерація 0**: 5 7 11 18 26 32 44 57 81 90 94 97 107 116 129 147 179.

**Ітерація 1**: 5 7 11 18 26 32 44 57 81 90 94 97 107 116 129 147 179.

**Ітерація 2:** 5 7 11 18 26 32 44 57 81 90 94 97 107 116 129 147 179.

Кожне порівняння зменшує число можливих кандидатів у 2 рази Максимальне число кроків пошуку буде в тому випадку, коли аргумент пошуку знаходиться на початку або в кінці масиву У цьому випадку буде потрібно log2n + 1 ітерацій. Дійсно, якщо число елементів у масиві дорівнює *п = 2т, ключ* буде знайдений, коли нерозглянутим залишиться тільки один елемент, тобто через *т* кроків У свою чергу, при заданому *п* маємо *т* = log2*n.* Після аналізу останнього елемента отримуємо загальне число ітерацій log2*n* + 1. Тому обчислювальна складність бінарного пошуку складає O (log2*n*)

Однак наведений алгоритм не дозволяє в загальному випадку точно вирішити завдання пошуку, коли файл або масив містять повторювані значення ключів. Розглянемо, наприклад, масив

5 7 11 18 26 32 44 57 81 90 94 97 107 129 129 147 179

в якому елемент *(Ключ)*129 міститься 2 рази Тоді, якщо аргумент пошуку буде дорівнює 129, алгоритм, як і колись, вкаже, що ключ знаходиться в 15-й позиції, тобто буде знайдено не перше, а друге значення ключа 129 (перше значення ключа розташоване в позиції 14) У ряді випадків ця помилка принципова, тоді вона усувається в результаті очевидною модифікації алгоритму бінарного пошуку.

**Порівняння послідовного і бінарного пошуку**

Нехай файл, в якому виконується пошук, відсортований і містить 1024 (210) елемента. У разі послідовного пошуку найбільше число ітерацій дорівнюватиме 1024, а бінарного 11. Тобто різниця в два порядки.

Порівняємо тепер тимчасові витрати на пошук в разі не відсортованого файлу. При послідовному пошуку максимальне число ітерацій, зрозуміло, зберігається. Бінарний пошук непридатний.

Виконаємо швидке сортування файлу. У середньому для цього буде потрібно 1024 \* log21024 = 10240 ітерацій. Далі виконаємо бінарний пошук, на який буде витрачено не більше 11 ітерацій.

Наведені цифри дозволяють зробити висновок: якщо файл невідсортований і в процесі обчислень завдання пошуку у файлі виникає порівняно рідко (у нашому прикладі не більше 10 разів), то можна застосовувати послідовний пошук, в іншому випадку більш доцільно перш впорядкувати файл або таблицю покажчиків і при обчисленнях застосовувати бінарний пошук. Як правило, саме такий підхід використовується на практиці.

У своїй програмі буду використовувати перший метод, а саме: “Метод Шинглів”, оскільки він найбільш простий для розуміння і найбільш точний, тому його і найчастіше використовують у програмах перевірки антиплагіату.

Для цього нам будуть потрібні файли, з якими будуть перевірятися документи(база данних) і сам документ, який потрібно перевірити.

Після завантаження файлу на перевірку, текст розбивається на шингли, де записується в масиви, які будуть порівнюватись. Оскільки робота з масивами клопітка і важка, для полегшення і оптимізації швидкодії коду програми будемо використовувати хеш.

**Хешування** (англ. Hashing) - перетворення масиву вхідних даних довільної довжини в (вихідну) бітовий рядок встановленої довжини, що виконується певним алгоритмом. Функція, яка втілює алгоритм і виконує перетворення, називається «хеш-функцією» або «функцією згортки». Вихідні дані називаються вхідним масивом, «ключем» або «повідомленням». Результат перетворення (вихідні дані) називається «хешем», «хеш-кодом», «хеш-сумою».

Наприклад, ми можемо подати на вхід 128-бітної хеш-функції роман Льва Толстого в шістнадцятковому вигляді або число 1. У результаті на виході ми в обох випадках отримаємо різні набори псевдовипадкових шістнадцятирічних цифр виду: «c4ca4238a0b923820dcc509a6f75849b».

При зміні вихідного тексту навіть на один знак, повністю змінюється результат хеш-функції.

Ця властивість хеш-функцій дозволяє застосовувати їх в наступних випадках:

* при побудові асоціативних масивів;
* при пошуку дублікатів в серіях наборів даних;
* при побудові унікальних ідентифікаторів для наборів даних;
* при обчисленні контрольних сум від даних (сигналу) для подальшого виявлення в них помилок (що виникли випадково або внесених навмисно), що виникають при зберіганні і / або передачі даних;
* при збереженні паролів в системах захисту у вигляді хеш-коду (для відновлення пароля за хеш-коду потрібно функція, яка є зворотною по відношенню до використаної хеш-функції);
* при виробленні електронного підпису (на практиці часто підписують не саме повідомлення, а його «хеш-образ»);

Далі цей хеш буде порівнюватись і після відпрацювання коду написаним на мові програмування PHP, буде виводитись на екран відсоток унікальності.

**PHP** (англ. *PHP: Hypertext Preprocessor* — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проект відкритого програмного забезпечення.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

PHP — мова, у код якої можна вбудовувати безпосередньо HTML-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (**<?php**) і продовжує виконання до того моменту, поки не зустріне закриваючий тег (**?>**).

В мові вбудовані бібліотеки для роботи з базами данних, такими як: MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase. Через стандарт відкритого інтерфейсу зв'язку з базами даних (Open Database Connectivity Standard — ODBC) можна підключатися до всіх баз даних, до яких існує драйвер.

У своїй програмі перевірки на антиплагіат було прийняте рішення використовувати СУБД MySQL.  
 **MySQL** — вільна система управління реляційними базами даних(СУБД). MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система управління базами даних (СУБД) з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Оболонкою графічного інтерфейсу при роботі з MySQL було вибрано **phpMyAdmin**.  
 **phpMyAdmin** — веб-додаток з відкритим кодом на мові PHP із графічним веб-інтерфейсом для адміністрування бази даних MySQL або MariaDB. phpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Ця програма користується великою популярністю у веб-розробників, оскільки дозволяє керувати базу даних MySQL без вводу SQL команд через дружній інтерфейс і з будь-якого комп'ютера під'єднаного до інтернету без необхідності встановлення додаткового програмного забезпечення.

На сьогоднішній день phpMyAdmin широко застосовується на практиці. Останнє пов'язано з тим, що розробники інтенсивно розвивають свій продукт, з огляду на всі нововведення СУБД MySQL. Переважна більшість українських провайдерів використовують цей застосунок як панель керування для того, щоб надати своїм клієнтам можливість адміністрування виділених їм баз даних. Ініціатором є німецький розробник Тобіас Ратшіллер (Tobias Ratschiller) і базується, так само як і phpPgAdmin, на аналогічному застосунку MySQL-Webadmin, що виник у 1997 році.

Для відображення графічного інтерфейсу сайту використовую гнучку тему адмін-панелі **Inspinia** написана на **HTML5, CSS3** і **Bootstrap3**.

**HTML** (англ. *HyperText Markup Language*) — стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді. HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документа та ідеологію структурної розмітки тексту. Попри те, що HTML — штучна комп'ютерна мова, вона не є мовою програмування.

HTML разом із каскадними таблицями стилів(CSS) та вбудованими скриптами — це три основні технології побудови веб-сторінок.  
  
 **CSS** (англ. *Cascading Style Sheets*) — спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

**Bootstrap** (початкова назва — *Twitter Blueprint*) — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків.

Bootstrap був розроблений Марком Отто та Джейкобом Торнтоном як фреймворк для забезпечення однаковості внутрішніх інструментів Twitter. Реліз із відкритим сирцевим кодом вийшов 19 серпня 2011 року.

Розглянемо детальніше даний проект. Оскільки в пріорітеті виконання завдання стоїть швидкість відпрацювання коду, тобто швидка перевірка файлу, реалізація даної роботи виконана на самописному фреймворку.

Для того щоб зрозуміти краще принцип роботи програмного коду, розглянемо детальну структуру папок і будову бази данних.

**Структура проекту**

Фреймворк побудований по принципу MVC, що робить його мобільним і швидким.

**MVC** (англ. *Model-view-controller*) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

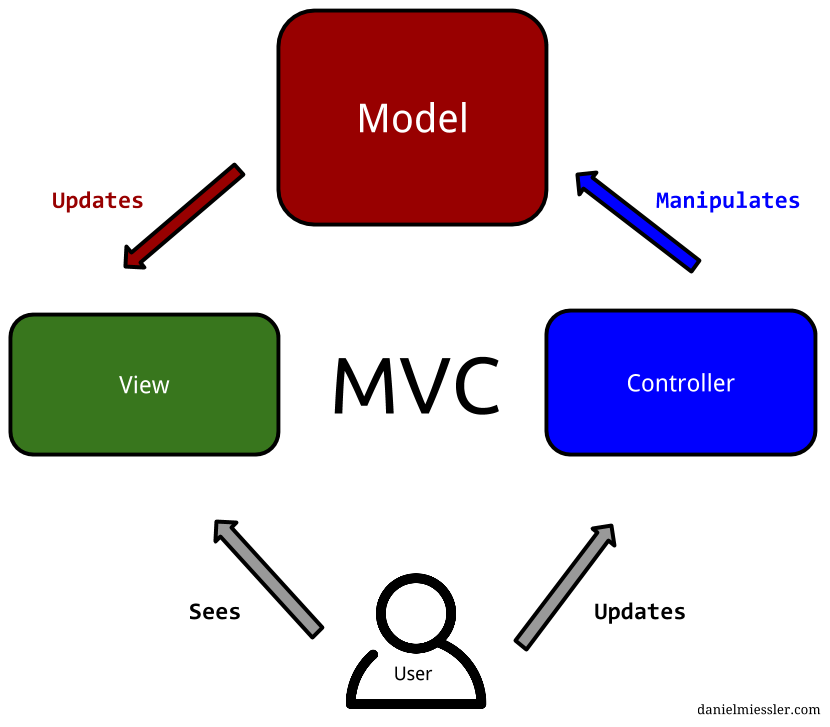
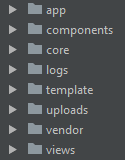


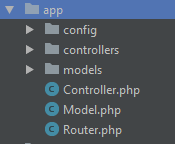
Рис. 1 Діаграма взаємодії між компонентами шаблону.

Програма включає в себе 9 папок, які утворюють основу програми: app, components, core, logs, temp, template, uploads, vendor, views.

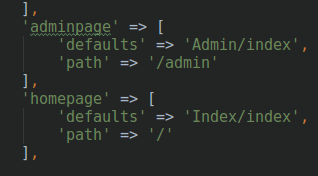
  
Рис. 1 Структура папок

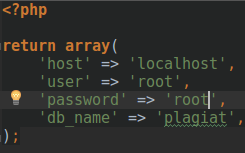
Розглянемо кожну з них:

**1. Папка app**

  
Рис. 1 App

У папці знаходяться **config, controllers, models** а також батьківські класи для контролерів(**Controller.php**), моделей(**Model.php**) а також маршрутизатор **Router.php**. Саме тут міститься основа фреймворку, адже без батьківських класів неможливе правильне функціонування інших файлів які унаслідують дані класи.  
 Також, тут знаходяться папки **config, controllers** i **models**. У **config** лежать конфігураційні файли: **db\_params.php,** де прописується підключення до бази даних(вказується хост, користувач, назва і пароль)і **routes.php**, де вказуються всі роути(шляхи) які будуть доступні в адресній строці браузера і назва контролера, що оброблятиме даний запит.

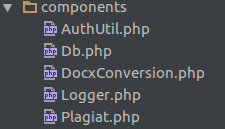
  
Рис1 routes.php

  
Рис. 1 db\_params

У **controllers** містяться контролери(логіка) програми, що відповідають за роботу з даними і їх обробку. Включає в себе **IndexController.php** - контролер, де описані методи сторінки перевірки файлу на плагіат. **AdminController.php** - методи адмін-панелі, де адмін може спостерігати і додавати файли у базу даних проекту. **SecurityController.php** - опис логіки login-page для входу в адмін-панель.

У **models** знаходяться моделі, у яких прописуються запити до бази даних і сортування вигляду самих даних. Включає в себе: **Session.php** і **User.php**, де дістаються з бази дані про користувача і сесію які створюються при створенні нового адміна. Завдяки правильній обробці в контролерах, інший користувач ніяк не зможе виконати вхід в адмін-панель. **Shingle.php** - модель, де описані запити для прийому шляху, за яким знаходяться файли у базі порівняння з вхідним документом.

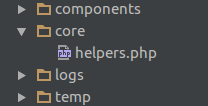
**2. Папка** **components.**

****Рис. 1 Components

У даній папці зберігаються класи, які виконують якийсь конкретний функціонал, а саме:

* **AuthUtil.php** - клас, де описана перевірка входу користувача в адмін-панель і дії в разі невдачі входу.
* **Db.php** - клас, що виконує підєднання до бази даних у методі *getConnections*, використовуючи параметри підключення із конфігураційного файлу *app/config/db\_params.php*.
* **DocxConvertion.php** - відбувається саме зчитування файлів, перевірка на правильність розширення і підготовка до подальшого використання.
* **Logger.php** – записує логи (службова та статистична інформація про події в системі) у файл по-замовчуванню, який задається у змінній *$logFile*, або ж у файл, який передається у функцію *log.* У нашому випадку системне логування здійснюється у папку **logs**.
* **Plagiat.php** - клас, методи якого очищають строки від розділових знаків, утворють шингли з тексту і підготовлюють до остаточного порівняння.

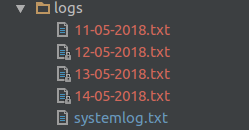
3. Папка **core**.

  
Рис. 1 Core

Папка являє собою ядро функцій користувача. У файлі **helpers.php** здійснюється опис функцій, таких як:

* dump() - виводить структуровану інформацію про змінну або вираз.
* path() - шлях для переходу користувача. Приймає 2 параметри: назву роута і масив із додатковими параметрами.
* redirect() - перенаправлення на url, який був вказаний.
* redirectToRoute() - перенаправення на вказаний роут із додатковими параметрами або без них.

4. Папка **logs**.

  
Рис. 1 Logs

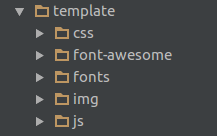
Дана папка містить 2 види текстових, службових файлів(логів), які пишуться паралельно з відпрацюванням коду:

* systemlog.txt - текстовий файл, куди зберігаються всі помилки відпрацювання скрипта. Адміністратор сайту, або відповідальна особа, може зайти в цей файл відстежити де код працює неправильно і виправити це негайно.
* d-m-y.txt(день-місяць-рік) - текстовий файл, який створюється при першій перевірці на даний день. Викладач який має доступ до проекту може відстежити в реальному часу, скільки співпадінь було знайдено із кожним файлом бази даних, і віднайти найбільший відсоток схожості тексту. Також, паралельно цю інформацію можна подивитись в адмін-панелі, де її надано у більш культурному вигляді.

5. Папка **temp**.

Недосвідчений користувач не побачить ніколи там ніякого вмісту. Ця папка слугує для тимчасового зберігання файлу, що перевіряється. Файл завантажується у папку, очищується від розділових знаків і прийменників, розбивається на шингли, порівнюється із файлами в базі, записується результат в log-файл, на екран виводиться відсоток найбільшого співпадіння, і після цього він видаляється. Саме тому ця папка завжди пуста.

6. Папка **template**.

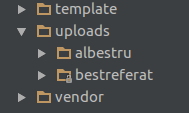
****Рис. 1 Template

Ця папка містить у собі файли для коректної роботи frontend частини даної програми.   
 Функціонал для адмін теми Inspinia:

* **css** - каскадні таблиці стилів для теми.
* **font-awesome** - стилі для шрифта який використовується на сайті.
* **fonts** - зберігаються, безпосредньо, самі шрифти.
* **js** - JavaScript файли які використовуються для гармонійного і динамічного користування(паралакси, слайдери, тостери і тд.)

Також, в цій директорії знаходиться папка **img**. Тут лежать всі картинки які використовуються на сайті. Коректно вказавши шлях її розташування, ми можемо вивести картинку на екран браузера.

7. Папка **uploads**.

  
Рис. 1 Uploads

**Uploads** включає в себе файли локальної бази даних, тобто **docx i pdf-документи**. Зайшовши в адмін-панель, модератор може завантажити конкретний файл або **zip-архів.** Після завантаження, архів розпаковується в дану директорію із збереженням всіх вкладеностей папок і їх назв.

Якщо хоча б один файл в архіві не буде відповідати заданим критеріям, папка не буде завантажена, і на екран буде виведена помилка.

9. Папка **views**.

  
Рис. 1 Views

У цій папці містяться html-сторінки сайту. По правилах структури MVC логіка і зовнішній вигляд сайту не повинні перемішуватись, саме тому **views** винесені в окрему папку. Кожен файл відповідає одній сторінці сайту. В даній директорії містяться папки:

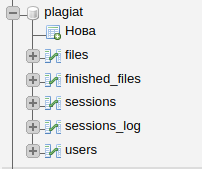
* **404 -** включає в себе файл **404.php**, який виводиться на екран в разі помилки відображення сайту(404 відповідь сервера)
* **admin** - сторінки адмінки. Містить папку **layouts**(фрагменти коду, які повторюються більше ніж на одній сторінці) і сторінки панелі: **admin.php**, де здійснюється завантаження нових файлів в базу, **files.php**, де адмін може переглянути кількість, назви файлів і шлях за яким вони лежать в локальній базі. **Logs.php** - виводяться в таблиці всі співпадіння які були знайдені під час перевірки файлу. Аналогія файла **/logs/d-m-y.txt**.
* **main -** файл головної сторінки **index.php**.
* **security -** містить файл **login.php**, який відповідає за відображення сторінки входу в адмін-панель.

Також в корені проекту лежать файли:

* .**gitignore** - оскільки під час виконання завдання, використовував репозиторій для зберігання проекту GitHub, цей файл зберігає список документів, які не потрібні при автоматичному завантаженні на даний веб-сервіс.
* **.htaccess** - конфігураційний файл налаштування сервера Apache.
* **composer.json -** файл конфігурацій PHP.
* **project.sql -** файл бази даних, який про виникненні несправностей можна завантажити в phpMyAdmin і відновити роботу сайту.
* **index.php -** файл основних налаштувань фреймворку і підключення системних модулів
* **read.me** - текстовий файл в якому описується принцим роботи даної програми.

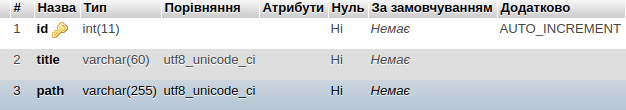
**Структура бази даних**

База даних, що має назву **plagiat**, складається із 5 унікальних таблиць.

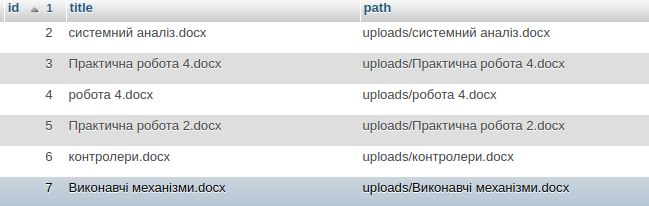
  
Рис. 1 Структура таблиць бази даних

Кожна з них відповідає за важливу частину роботи, яку розглянемо нижче:

* **files -** таблиця для зберігання шляху до тимчасових файлів, що завантажуються на перевірку. Дана таблиця відповідає за “читання” файлу, адже без шляху за яким лежить файл і його назвою, програма не могла б віднайти його для подальшого розбиття на шингли і порівняння із іншими файлами. Таблиця містить 3 поля:
  + **id -** первинний ключ елемента, по якому відбувається витягнення з бази. Тип даних: int() - цілі числа від 0 до 2 147 483 647. AUTO\_INCREMENT забезпечує генерування унікального ідентифікатора для нових елементів таблиці.
  + **title -** назва файлу що завантжується на перевірку. Тип даних: varchar(60) - строковий тип, здебільшого для зберігання тексту. Дозволене максимальне значення строки: 60 символів. Utf-8\_unicode\_ci - спеціальне кодування тексту для букв кирилиці.
  + **path** - зберігає шлях до файлу завантаження. Тип даних: varchar(255) - строковий тип, максимальної довжини. Дозволене максимальне значення строки: 255 символів.

  
Рис. 1 Таблиця Files

* **finished\_files** - таблиця для зберігання шляху і назви файлів у локальній базі. По структурі така ж як і **files**.

  
Рис. 1 Таблиця Fineshed\_files

У таблицю записуються нові значення, коли адмін завантажує нові файли через адмін-панель. Саме за цими значеннями відбувається перевірка файлу.

* **sessions** - таблиця, в якій зберігаються дані про користувача що ввійшов в адмін-систему.
  + **id** - первинний ключ елемента, тип**:** int().
  + **user\_id** - ключ із таблицею користувача, тип: int().
  + **ip -** відбувається запис ip-адреси юзера, тип: varchar(10). Максимально дозволена довжина строки: 10 символів.
  + **user\_agent** - записуються дані про браузер та операційну систему, що використовує користувач. Тип: text, адже ця строка може бути довільної довжини.
  + **hash -** перетворення паролю користувача в хеш для неможливості входу зловмисного користувача і злому сторінки. Тип: varchar(60). Максимальне значення строки:60 символів.
  + **ext\_time** - записується дата останнього входу в адмін-панель. Дані для перевірки на правдивість користувача. Оскільки ці дані можна підробити, важливо хоча б приблизно знати межі в яких користувач отримував доступ до панелі. Тип: datetime.

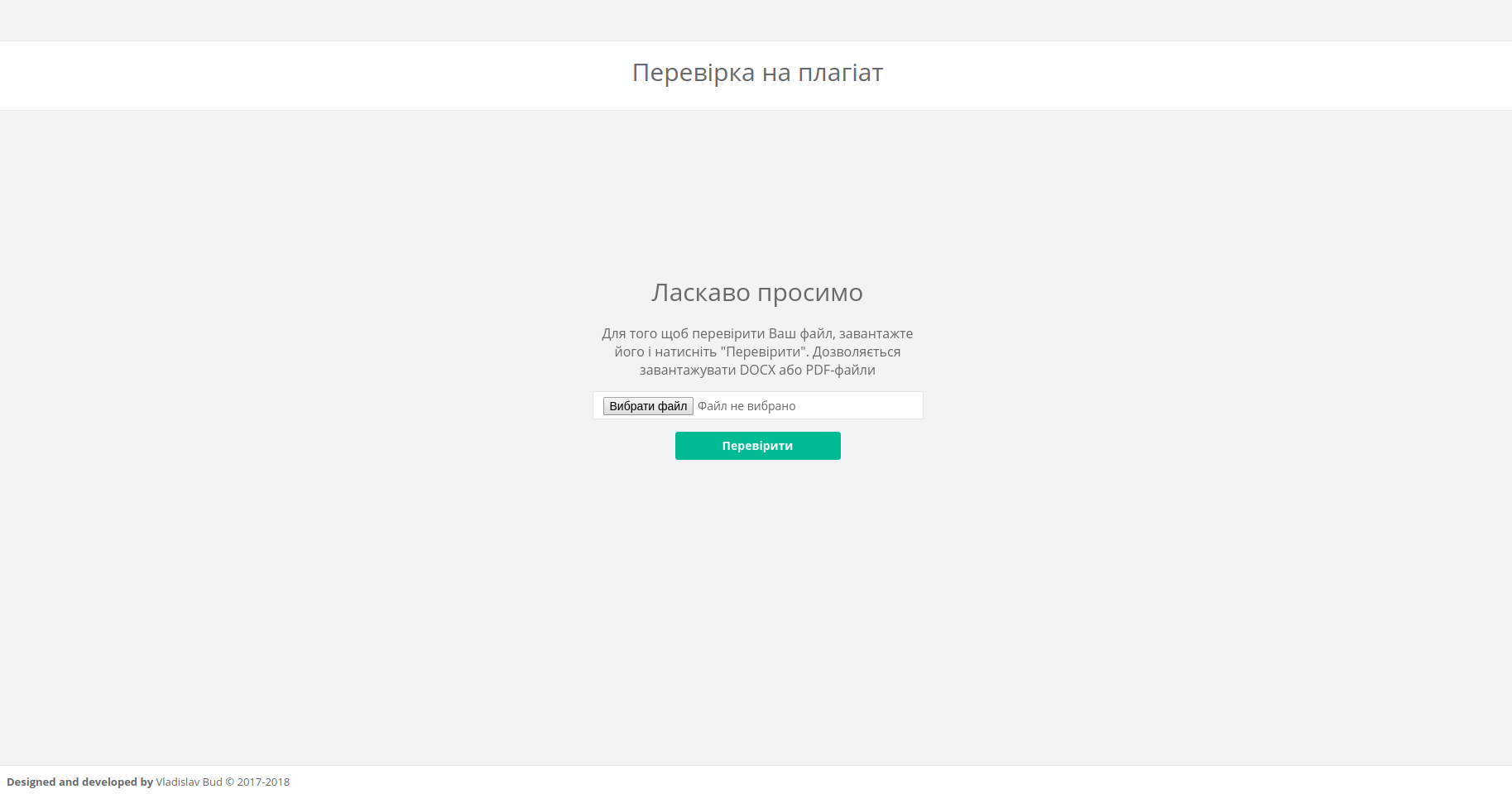
  
Рис. 1 Таблиця Sessions

Завдяки полям **ip, user\_agent і hash** відбувається перевірка на правдивість даних про користувача. Якщо хоча б одне із цих даних не збігається користувач ніколи не зможе проникнути на сайт адмін-панелі, що захищає від зловмисних атак. Навіть якщо зловмисник зможе підробити ip-адрес, що не є досить складним для досвідченого фахівця, він ніколи не зможе обійти перевірку на user\_agent i hash, тим більш спробувати прорахувати ці дані. Хеш створюється при реєстрації юзера, це строка із 16 довільних символів, що включає в себе і цифри і букви. Дізнатись ці дані без доступу до бази, неможливо.

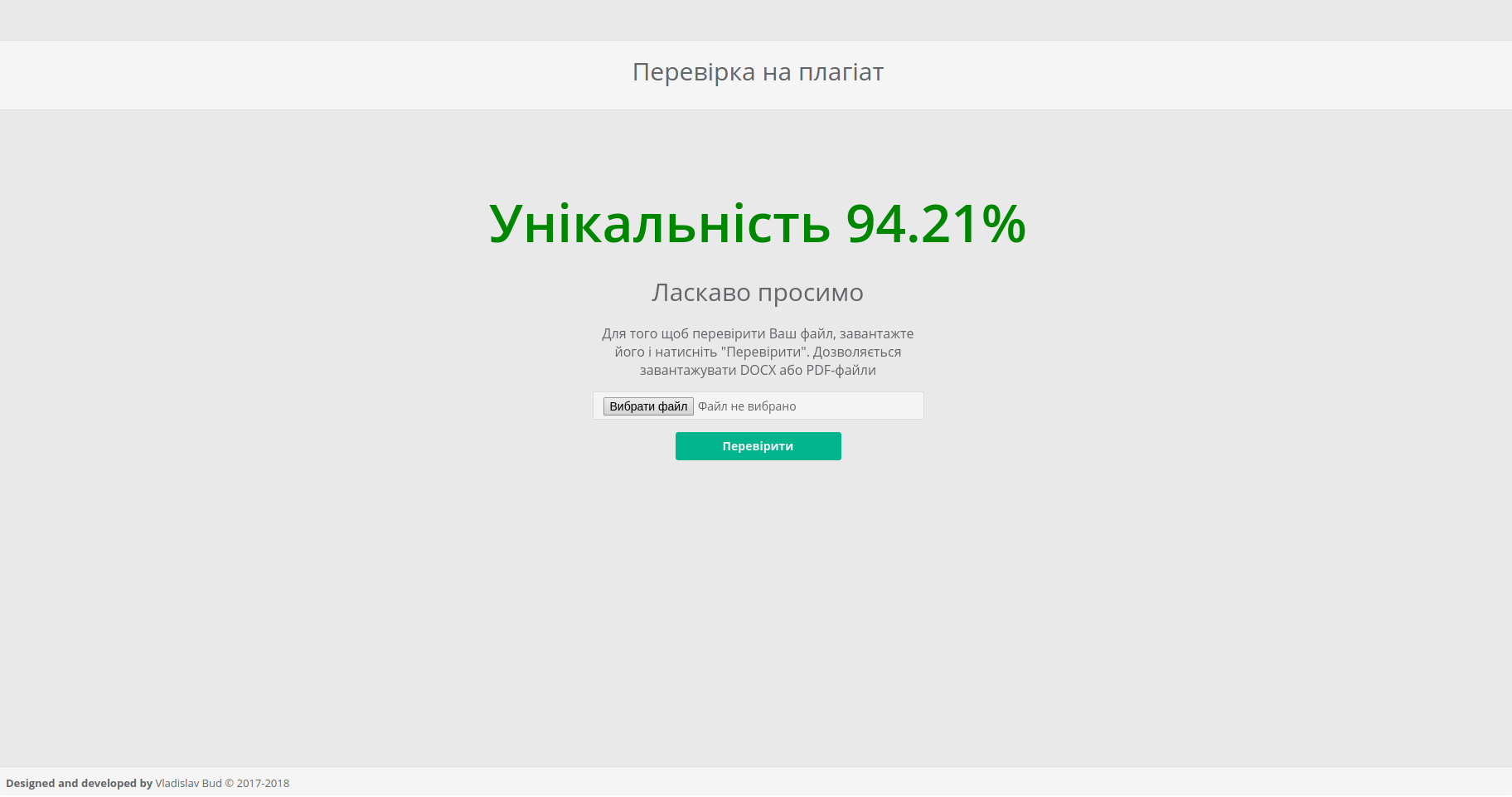
* **sessions\_logs** - таблиця, яка відповідає за запис помилок при створенні користувача. Структура аналогічна **sessions.** В разі хорошого відпрацювання коду дані пишуться в таблицю **sessions,** в разі помилок у **sessions\_logs.**
* **users -** таблиця, що зберігає дані про користувача.
  + **id -** первинний ключ елемента, тип**:** int().
  + **username** - записується login користувача.Тип: varchar(32). Максимальна довжина строки: 32 символа.
  + **password** - зберігається хеш пароля користувача. Тип: varchar(32). Максимальна довжина: 32 символа.
  + **created\_at** - вносяться дані коли був зареєстрований користувач у системі. Тип: datetime. Записується у вигляді: рік-місяць-день, години-хвилини-секунди.

  
Рис. 1 Таблиця Users

Розглянемо даний проект на практиці: спробуємо перевірити файл на плагіат. Відкриємо браузер і введемо в адресній строці назву сайту “[**http://project.loc/**](http://project.loc/)”. Після завантаження ми опиняємось на головній сторінці.

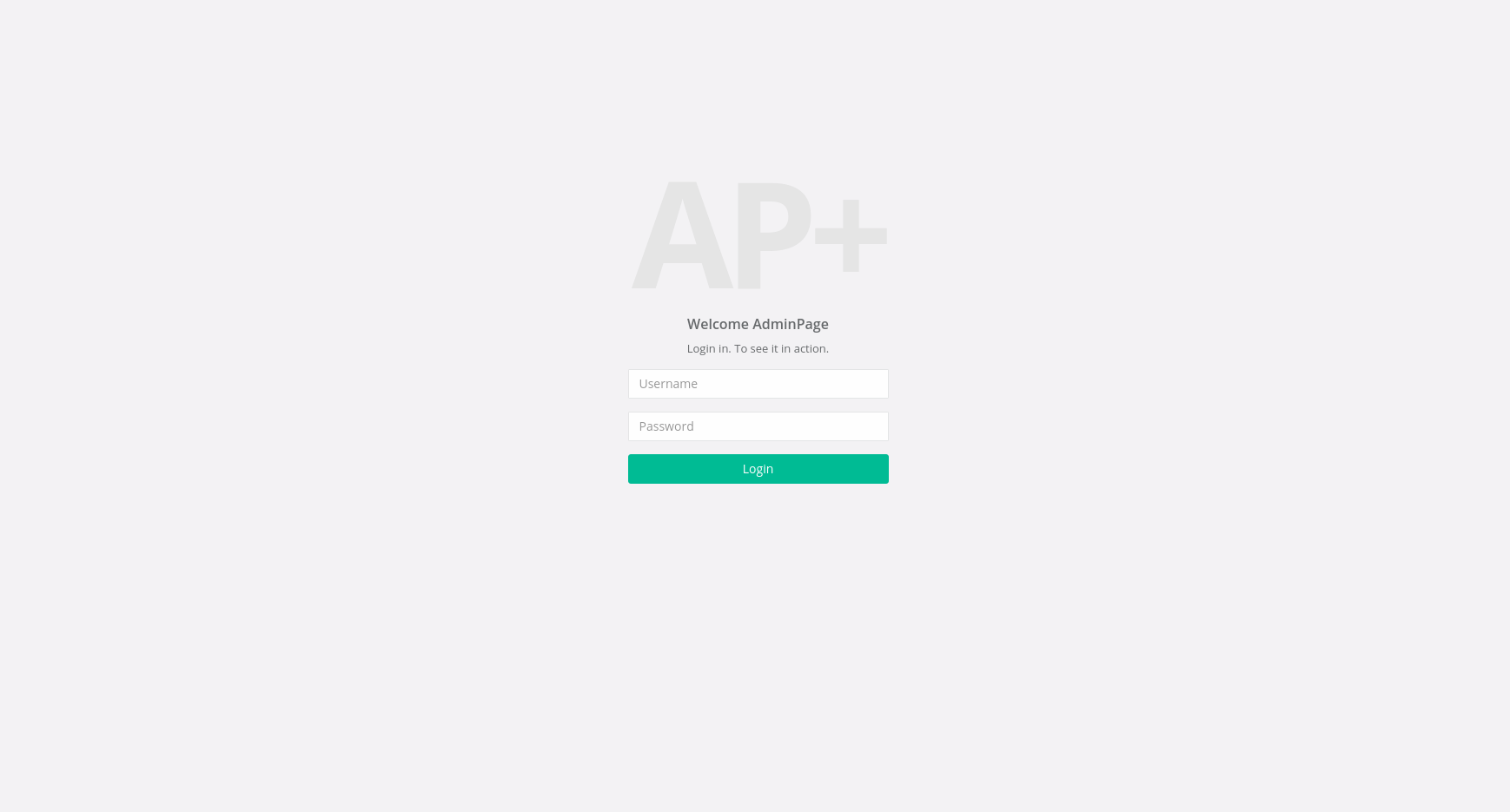
  
Рис. 1 Вигляд головної сторінки сайту

Завантажимо файл, якого немає у нашій локальній базі даних, і який має розширення **.docx** або **.pdf.** Відсоток унікальності тексту виводиться у вигляді строки над формою вводу. Зачекавши декілька секунд**,** отримаємо такий результат відпрацювання коду:

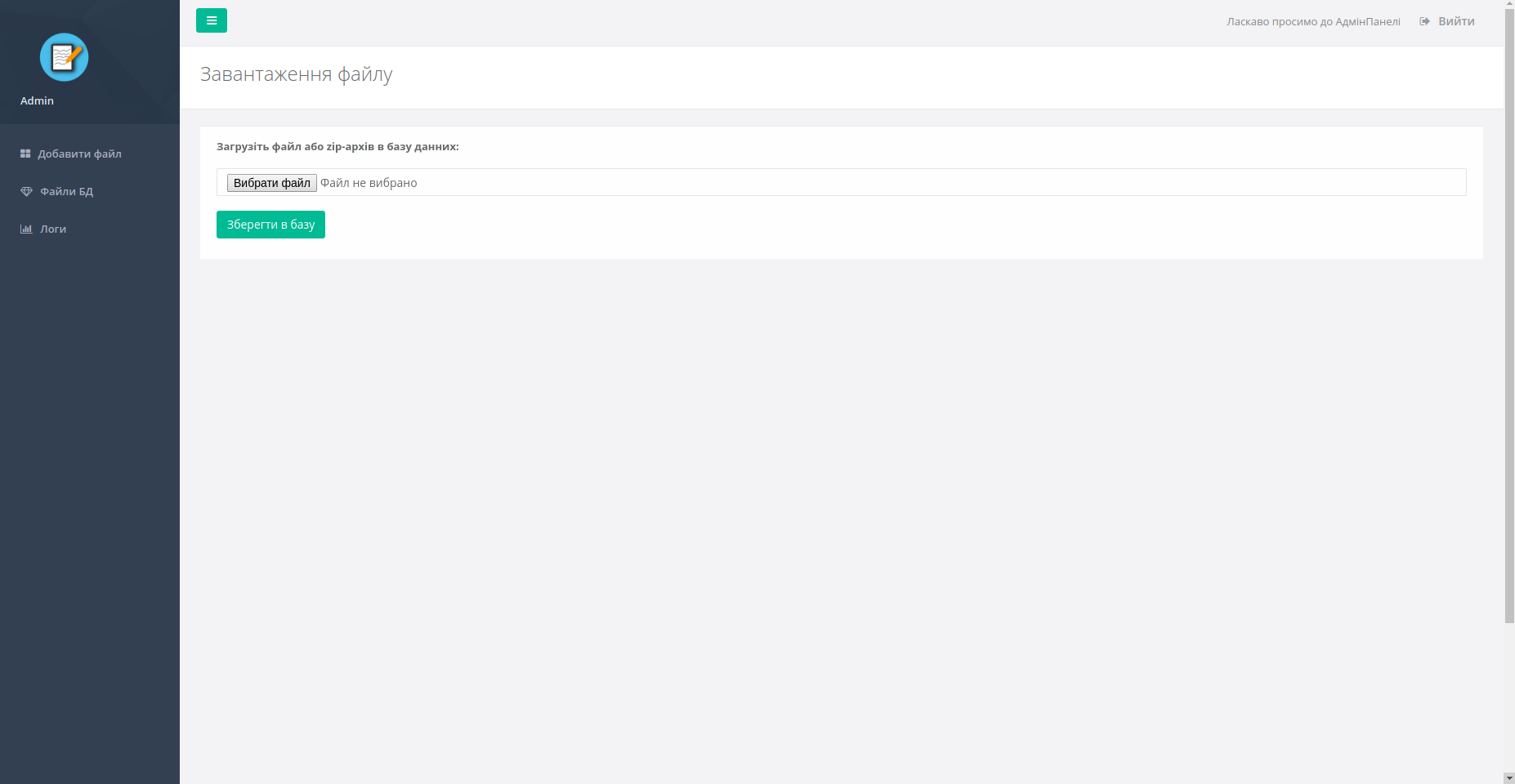
  
Рис. 1 Вивід відсотку унікальності.

Вибраний для перевірки файл має 5.79% копірайту. Унікальність тексту становить 94.21%, що є дуже хорошим результатом.

Тепер перевіримо логи у адмін-панелі і txt-файл логів даного дня. Для цього перейдемо по шляху “**/admin**”. Оскільки ми не являємось автентифікованим користувачем нас перенаправило на сторінку “**/login**” введення логіну і паролю.

  
Рис. 1 Сторінка входу в адмінку

Ввівши коректні дані, користувач отримує доступ до адмін-панелі.

  
Рис. 1 Вигляд сторінки адміністрування сайту

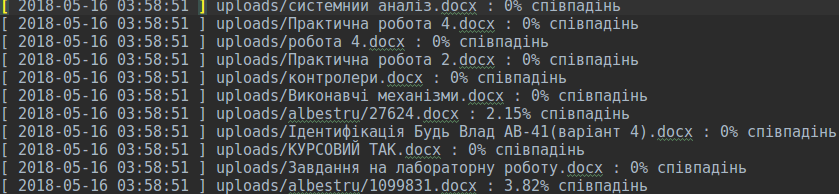
З лівої сторони знаходиться головне меню. Перейдемо на сторінку логів. Бачимо сформовану таблицю відносно всіх файлів у базі даних.

  
Рис.1 Таблиця логів

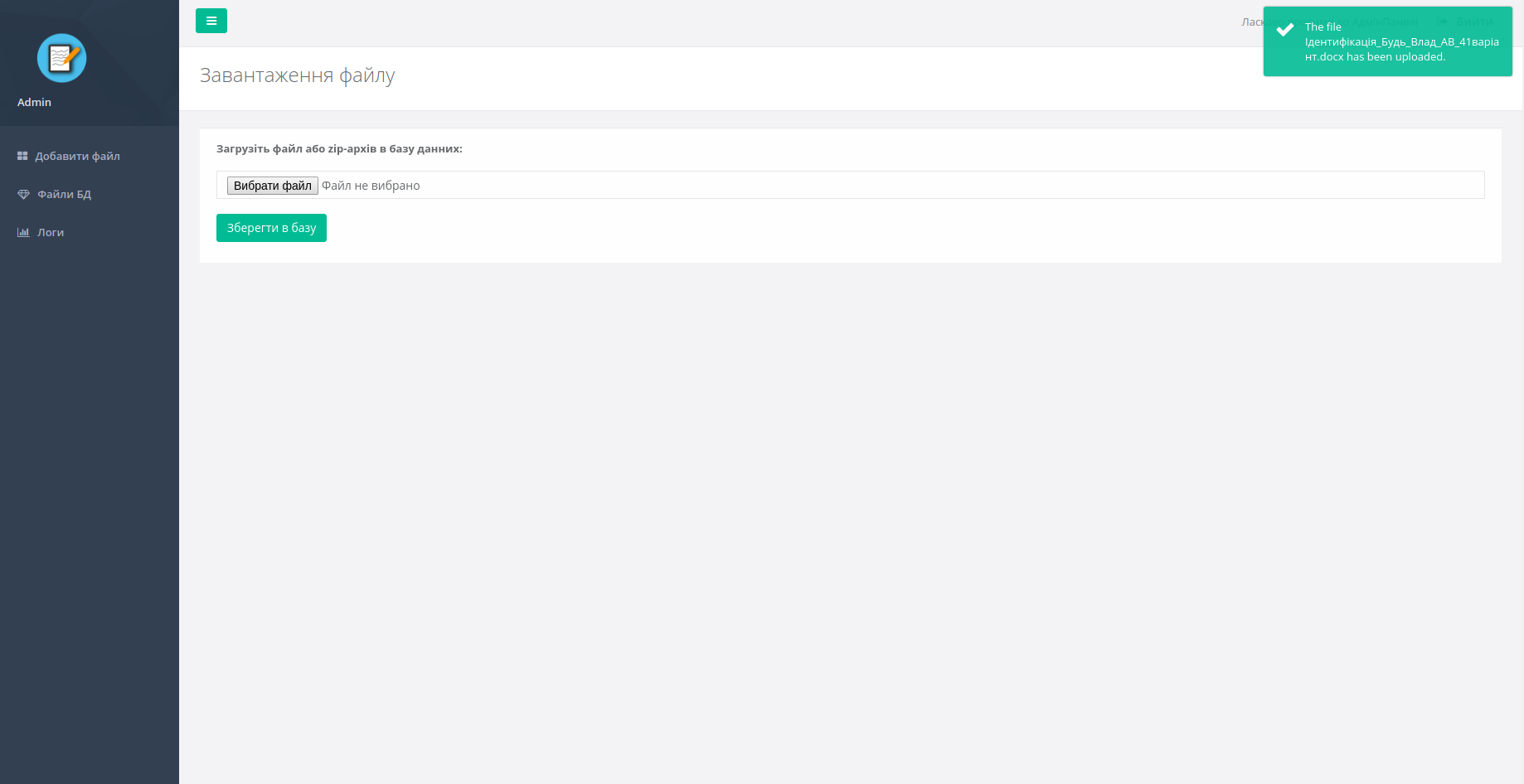
Для швидшого пошуку елементів таблиці присутнє поле пошуку, сортування даних по вибраному полю і вибір кількості відображення записів на даній сторінці. Якщо кількість елементів перевищує максимальну кількість записів на даній сторінці створюється пагінація. Весь цей функціонал реалізований, використовуючи **DataTables**.

**DataTables** являє собою плагін для покращення таблиць використовуючи бібліотеки jQuery, додаючи можливості сортування, підбору пріоритету та фільтрації звичайних HTML-таблиць з мінімальними зусиллями.

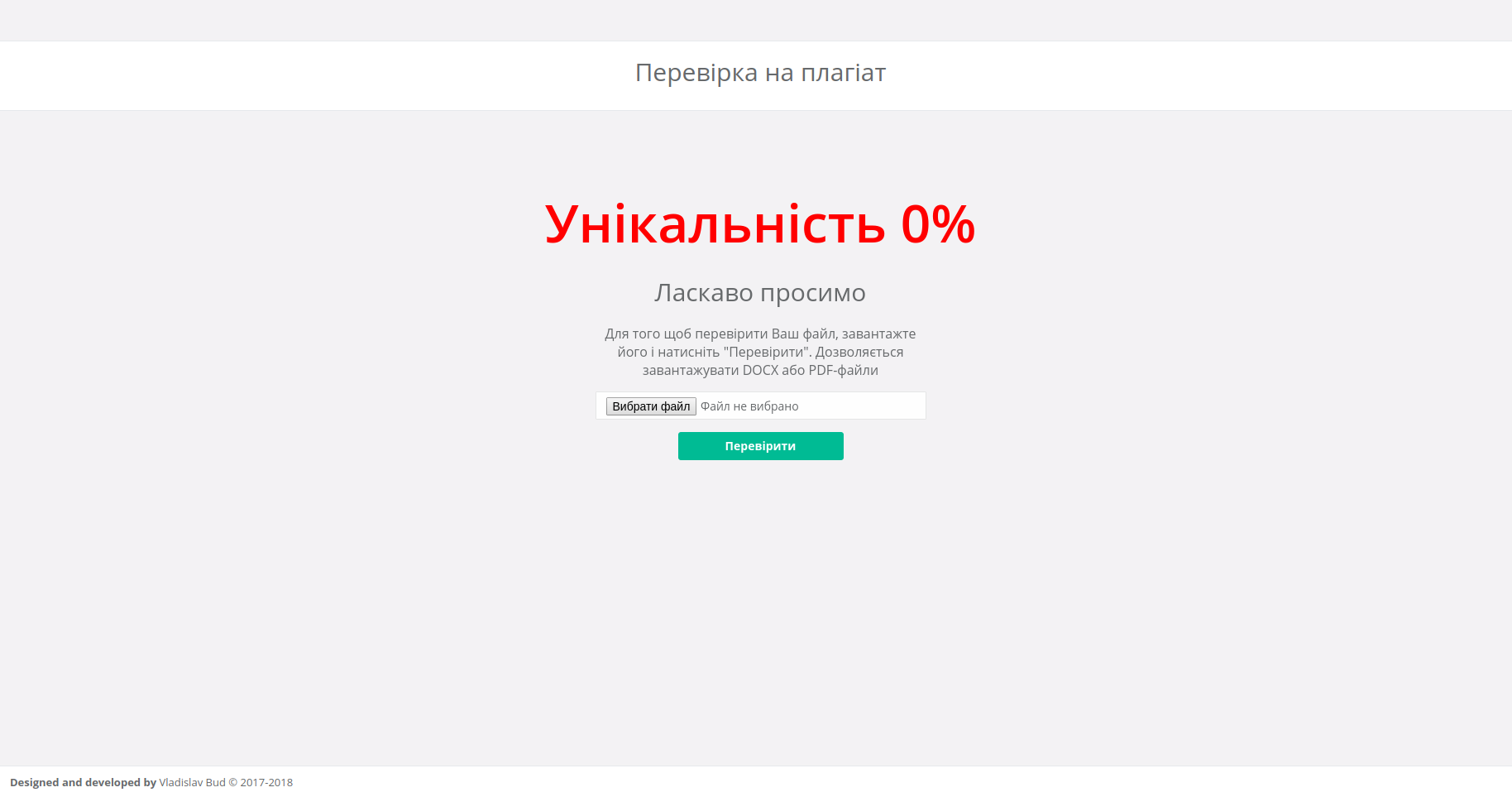
Ті ж самі дані ми можемо побачити у лог-файлі:

  
Рис. 1 16-05-2018.txt

Завантажимо даний файл в базу даних і проаналізуємо чи зміниться результат перевірки. Для цього повернемося на головну сторінку керування сайтом і завантажимо цей файл у базу даних. Про успішне завантаження нас повідомить відповідь програми:

  
Рис.1 Додавання файла в базу даних

Повернемось на головну сторінку сайту і спробуємо завантажити файл повторно. Після успішного завантаження програма перевірки плагіату видає інший результат:

  
Рис. 1 Відповідь програми після завантаження файлу в базу даних.

Ми бачимо, що унікальність даного тексту становить 0%, тобто плагіат = 100%, адже копія цього файлу знаходиться в папці із документами.

Повну структуру даного проекту і його базу данних ви можете переглянути і завантажити на репозиторії **GitHub** за посиланням: “**https://github.com/VladBud/plagiat**”.

**Використана література**

**Програмне забезпечення для перевірки наукових текстів на плагіат: інформаційний огляд** /

автори-укладачі: А. Р. Вергун, Л. В. Савенкова, С. О. Чуканова ;

редколегія: В. С. Пашкова, О. В. Воскобойнікова-Гузєва, Я. Є. Сошинська ;

Українська бібліотечна асоціація. – Київ : УБА, 2016.

Гельфанд І. М. Лекції з лінійної алгебри. - 5-е. - М .: Добросвет, МЦНМО, 1998. - 319 с. - ISBN 5-7913-0015-8.

Гельфанд І. М. Лекції з лінійної алгебри. 5-е изд. - М .: Добросвет, МЦНМО, 1998. - 320 с. - ISBN 5-7913-0016-6.

Кострикін А. І., Манін Ю. І. Лінійна алгебра і геометрія. 2-е изд. - М .: Наука, 1986. - 304 с.

Manber, Udi, Finding Similar Files in a Large File System, USENIX Winter Technical conference, January, 1994, CA.

Каплунов Д. Копирайтинг массового поражения. — М.: «Питер», 2013. — 256 с. — ISBN 978-5-459-00680-3.

Слободянюк Є. Настольная книга копирайтера. — М.: «Манн, Иванов и Фарбер», 2011. — 216 с. — ISBN 978-5-91657-167-7.

Бартеньев О. В. 1С: Підприємство: програмування для всіх. Базові обєкти та розрахунки на одній дискеті М: Діалог-МІФІ, 2005 464 с.