Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(освітньо-кваліфікаційний рівень)

ГЮИК.509000.005 ПЗ
(позначення документа)

Розробка методу використання технологій Big Data та Data Mining в інтелектуальних системах обробки неструктурованих даних
(тема)

Виконав:
студент *2* курсу, групи *СПРм-18-2*
Демченко О. Е.
(прізвище, ініціали)

Спеціальність *122 – Комп'ютерні науки*
(код і назва напряму, спеціальності)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва освітньої програми)

Керівник *проф. Ситніков Д. Е.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *проф. Гребеннік І.В.*
(підпис) (посада, прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*
(повна назва)

Кафедра *Системотехніки*
(повна назва)

Рівень вищої освіти *другий (магістерський)*
(освітньо-кваліфікаційний рівень)

Спеціальність *122 – Комп'ютерні науки*
(код і повна назва)

Тип програми *освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Системне проектування*
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Демченко Олександру Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка методу використання технологій Big Data та Data Mining в інтелектуальних системах обробки неструктурованих даних*

затверджена наказом по університету від « » 2020 р. №_____

2. Термін подання студентом роботи до екзаменаційної комісії 2020 р.
3. Вихідні дані до роботи *Функція: Розробка методу використання технологій Big Data та Data Mining в інтелектуальних системах обробки неструктурованих даних. Форма діалогу: веб-сервіс. Перелік використовуваних програмних засобів: ОС Windows, інтегроване середовище розробки Visual Studio, C#, .NET Core, TypeScript, MongoDB. Технічне забезпечення: комп'ютер з веб-браузером Google Chrome версії вище за 49, Firefox версії вище за 55 незалежно від операційної системи*
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ 4.2 Аналіз предметної області і постановка задачі 4.2 Аналіз предметної області 4.3 Опис та аналіз сучасного стану проблеми 4.4 Аналіз існуючих систем для збору даних 4.5 Аналіз існуючих інструментів Big Data 4.6 Постановка задачі 4.7 Аналіз існуючих методів пошуку асоціативних правил 4.8 Алгоритм AIS 4.9 Алгоритм SETM 4.10 Алгоритм Apriori 4.11 Опис структури та роботи системи 4.12 Опис структури блока збору інформації 4.13 Опис структури блока обробки інформації 4.14 Загальний опис роботи системи 4.15 Проектування програмного засобу 4.16 Діаграма використання 4.17 Бізнес ролі в системі 4.18 Вимоги до програмного додатка 4.19 Проектування інтерфейсу користувача 4.20 Розробка програмного засобу 4.21 Обґрунтування вибору мов

програмування 4.22 Обґрунтування вибору архітектури 4.23 Обґрунтування вибору СУБД 4.24 Моделювання даних системи 4.25 Шаблони доступу до даних 4.26 Захист інформації в системі 4.27 Розгортання системи 4.28 Екранні форми додатка 4.29 Експерименти в обробці даних 4.30 Висновки .

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів,комп'ютерних ілюстрацій (слайдів). *5.1 Алгоритм Apriori 5.2 Алгоритм створення форми 5.3 Алгоритм створення колекції форм 5.4 Алгоритм відповіді на колекцію форм 5.5 Алгоритм запуску аналізу колекції форм 5.6 Діаграма прецедентів для користувача «Гість» 5.7 Діаграма прецедентів для користувача «Користувач» 5.8 Діаграма прецедентів для користувача «Адміністратор» 5.9 Узагальнена схема роботи Web API 5.10 Час прогону алгоритма Apriori 5.11 Використання CPU алгоритмом Apriori 5.12 Використання оперативної памяті алгоритмом Apriori*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання «30» березня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Отримання завдання на дипломне проектування	30.03.2020	
2.	Аналіз завдання, пошук літератури та аналогів з теми дипломної роботи	31.03-05.04.2020	
3.	Опрацювання літератури та аналіз об'єкту дослідження	06.04 -11.04.2020	
4.	Формування постановки задачі	12.04.2020	
5.	Проектування сервісу	13.04-19.04.2020	
6.	Розробка та тестування програмного засобу	20.04-29.04.2020	
7.	Аналіз результатів, отриманих за допомогою програмного засобу	30.04.2020	
8.	Оформлення пояснювальної записки	01.05.2020-14.04.2020	
9.	Представлення на рецензування	15.05.2020	
10.	Представлення дипломної роботи	23.05.2020	

Студент _____
(підпис)

Керівник роботи _____ *проф. Ситніков Д. Е.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Атестаційна робота містіть: 77 сторінки, 51 рисунка, 3 таблиці, 4 додатків, 77 джерел. Графічна частина атестаційної роботи містіть 12 плакатів.

Об’єктом дослідження є процес збору та обробки інформації за допомогою Big Data та Data Mining.

Предметом дослідження є методи збору інформації та методи пошуку асоціативних правил.

Мета дослідження – проаналізувати існуючі методи та підходи до реалізації систем збору та обрабки інформації, провести порівняльний аналіз обраних методів та алгоритмів, розробити прототип системи збору та обробки інформації та виконати аналіз отриманих результатів.

Методи дослідження – системний підхід, методи Big Data та Data Mining.

Галузь застосування – реалізований продукт може використовуватись спеціалістами з маркетингу та реклами, керівниками фірм та організацій, HR спеціалістами, вчителями та викладачами. Також подібний програмний засіб може бути використаним для наукових досліджень з питань психології (соціальні опитування) .

ONLINE, ФОРМА, КОЛЕКЦІЯ ФОРМ, ООП, СУБД, MONGODB, C#, ANGULAR, API, RULES, APRIORI, BIG DATA, DATA MINING, AZURE, APP SERVICE, FUNCTION

ABSTRACT

Thesis consists of: 77 pages, 51 figures, 3 tables, 4 applications, 77 sources. The graphic part of the certification work is 12 posters.

The object of research is the process of collecting and processing information using Big Data and Data Mining.

The subject of research is methods of collecting information and methods of searching for associative rules.

The purpose of the study is to analyze existing methods and approaches to the implementation of information collection and processing systems, to conduct a comparative analysis of selected methods and algorithms, to develop a prototype of information collection and processing system and to analyze the results.

Research methods - system approach, Big Data and Data Mining methods.

Scope - the sold product can be used by marketing and advertising specialists, managers of companies and organizations, HR specialists, teachers and lecturers. Also, such software can be used for research in psychology (social surveys).

ONLINE, FORM, FORM COLLECTION, OOP, DBMS, MONGODB, C#, ANGULAR, API, RULES, APRIORI, BIG DATA, DATA MINING, AZURE, APP SERVICE, FUNCTION

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

СУБД – система управління базами даних.

CASE – Computer Aided Software Engineering (сукупність методів та засобів автоматизованого проектування інформаційних систем систем);

CLI - Command-line Interface – різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд).

DI – Впровадження залежності (Dependency injection, DI) — шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» для отримання залежностей.

DRY – do not repeat yourself (не повторюй себе).

HTML – HyperText Markup Language (мова гіпертекстової розмітки);

MS – Microsoft;

MVC – Model-View-Controller (архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення);

YAGNI – you are not gonna need it (це не знадобиться).

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ 8
1.1 Аналіз предметної області 8
1.1.1 Аналіз питання онлайн опитування 8
1.1.2 Аналіз питання обробки даних 12
1.2 Опис та аналіз сучасного стану проблеми 13
1.3 Аналіз існуючих систем для збору даних 15
1.4 Аналіз існуючих інструментів Big Data 21
1.5 Постановка задачі 23
2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПОШУКУ АСОЦІАТИВНИХ ПРАВИЛ 25
2.1 Алгоритм AIS 25
2.2 Алгоритм SETM 26
2.3 Алгоритм Apriori 27
3 ОПИС СТРУКТУРИ ТА РОБОТИ СИСТЕМИ 29
3.1 Опис структури блока збору інформації 29
3.2 Опис структури блока обробки інформації 29
3.3 Загальний опис роботи системи 31
4 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ 36
4.1 Діаграма використання 36
4.2 Бізнес ролі в системі 39
4.3 Вимоги до програмного додатка 39
4.4 Проектування інтерфейсу користувача 40
5 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ 48
5.1 Обґрунтування вибору мов програмування 48
5.1.1 Back-End 48
5.1.2 Front-End 49
5.2 Обґрунтування вибору архітектури 50
5.2.1 Архітектура Back-End 50
5.2.2 Function App 52
5.2.3 Взаємодія Back-End та Front-End 53
5.3 Обґрунтування вибору СУБД 54
5.4 Моделювання даних системи 55
5.5 Шаблони доступу до даних 57
5.6 Захист інформації в системі 58
5.7 Розгортання системи 59
5.8 Екранні форми додатка 60
5.9 Експерименти в обробці даних 70
ВИСНОВКИ 72
ПЕРЕЛІК ПОСИЛАНЬ 73
Додаток А Графічний матеріал атестаційної роботи 79

Додаток Б Текст програми..... 94
Додаток В «Специфікація»..... 118
Додаток Г «Відомість атестаційної роботи» 120

ВСТУП

Мабуть, кожен з нас, навіть ненароком, зіткався з проблемою опитування. Опитування студентів, друзів та, можливо, навіть працівників. Якщо опитати близько 20 чоловік не є великою проблемою, то що робити, коли у тебе є власний бізнес з великою кількістю клієнтів? Ви хочете дізнатися, що відволікає ваших відвідувачів? Що їх дратує? Якої інформації їм не вистачає для того, щоб зробити покупку? На допомогу приходять маркетингові агентства. Але, чи завжди виправдані значні витрати на проведення досліджень за допомогою маркетингових агентств, і чи є альтернатива агентствам взагалі? Безумовно, для проведення глибокого аналізу ринку або порівнянних за масштабом завдань, потрібно звернутися до маркетологів, якщо, звичайно, дозволяє бюджет. Але в дослідженнях менш глобальних допоможуть онлайн опитування.

Проведення опитувань клієнтів - один з ключових етапів побудови успішної стратегії по оптимізації конверсії. Існують сервіси, що дозволяють провести онлайн анкетування максимально просто: ви створюєте опитування, поширюєте його і отримуєте вже оброблені результати.

Але, чи здатні вони справитися з великою кількістю даних та чи завжди можна довіряти стороннім сервісам? Адже ми не можем бути впевнені, що наші дані достатньо добре захищені і що не існує можливості передачі їх конкурентам, або навіть шахраям.

Метою атестаційної роботи являється розробка програмного засобу, який міг би допомогти в вирішенні поставленого питання.

Отже, цілю даної роботи є програмний засіб, який використовується швидкого збору та обробки отриманої інформації. Цільова аудиторія користувачів такого програмного засобу – спеціалісти з маркетингу та реклами, керівники фірм та організацій, HR спеціалісти, вчителі та викладачі. Також подібний програмний засіб може бути використаним для наукових досліджень з питань психології (соціальні опитування) .

Засіб повинен виконувати такі задачі: створювати форми, які в сукупності будуть утворювати колекції форм, розповсюдження опитувань, швидкий доступ до результатів, можливість їх аналізу та експорту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

У зв’язку з швидким та масштабним розвитком цифрової електроніки, усі сфери повсякденного життя та бізнесу переходять у електронний або віддалений формат зв’язку. Комунальні послуги, покупки в магазинах и т.д. – усі ці дії сьогодні відбуваються за допомоги цифрових пристроїв, таких як комп’ютер або смартфон. Людям зручніше натиснути декілька кнопок на своєму смартфоні, а не стояти в нескінченних чергах та витрачати свій час.

В сучасному суспільстві використання різних пристроїв для взаємодії з навколишнім світом, а також іншими людьми є не від’ємною частиною повсякденного життя кожної людини. Більшість сфер які будь-яким чином взаємодіють з людиною намагаються використовувати інформаційні технології задля покращення якості свої послуг та інше.

Те саме стосується збору і обробки даних. Адже, набагато легше відкрити сторінку в інтернеті та швидко відповісти на поставлені питання, ніж заповнювати величезну кількість паперових анкет.

1.1.1 Аналіз питання онлайн опитування

Онлайн-опитування (інтернет-опитування, web-опитування) – це метод збору соціологічної інформації, який здійснюється на основі використання інтернет-технологій. Найчастіше онлайн-опитування проводяться по анкеті, розміщеній на сайті і заповнюється з комп'ютера або мобільного пристрою в режимі онлайн [1].

Першим зареєстрованим опитуванням, здійсненим за допомогою віддаленої взаємодії (електронної пошти), вважається дослідження Kiesler і Sproull (1986) [2], присвячене вивченню e-mail-комунікації в рамках компанії.

Метод онлайн-опитування розвивався від використання електронної пошти до створення онлайн-панелей. Опитування за допомогою розсилки анкет по електронній пошті (e-mail-опитування) відрізнявся простотою і не вимагав спеціальної комп'ютерної підготовки від дослідника. Спочатку обробка повернутих анкет йшла вручну, пізніше програмне забезпечення дозволило обробляти анкети напівавтомат. Найбільше опитування по електронній пошті підійшли для локальних досліджень,

наприклад, в рамках установ. Опитування великих аудиторій проводився в групах новин (newgroups), які створювалися для обговорення певних тем. Текст анкети розміщувався на сайті і поновлювався раз в кілька днів, щоб його могли бачити нові відвідувачі і вводити свої відповіді. Метод дозволяв опитувати цільові групи, які складно залучити до опитувань (наприклад, анонімних алкоголіків).

Методологічні дискусії щодо використання онлайн-опитувань також стали активно вестися починаючи з середини 1990-х. Вказувалося, що онлайн-опитування можуть охопити максимально доступну вибірку за мінімальною ціною і в найкоротші терміни в порівнянні з будь-якими іншими з раніше існуючих форм проведення досліджень (Anderson & Gansneder 1995 [3]; O'Lear 1996 [4]; Kehoe & Pitkow 1996 [5]; Schmidt 1997 [6]; Smith 1997 [7]). Sheehan & McMillan (1999) [8] встановили, що опитувальник, надісланий респонденту звичайною поштою, повертається заповненим в середньому через 11,8 днів, в той час як при електронній формі відправки термін повернення скорочується до 7,6 днів. Інші дослідники, наприклад, Harris (1997) [9] повідомляли про ще більш стислі терміни реакції на опитування по електронній пошті (в середньому 2-3 дні).

Dillman (2000) [10], відзначаючи, що онлайн-опитування грунтуються на двох попередніх змінах в сфері проведення досліджень, а саме з випадковою вибіркою респондентів (1940-ті роки) і впровадженні телефонних опитувань (1970-ті роки), називав їх найістотнішим методологічним перетворенням дослідницької діяльності. Couper (2000) [11], в свою чергу, підкреслював, що соціальні наслідки поширення онлайн-опитувань настільки ж значні як і методологічні: доступність і низька ціна технологій роблять можливим проведення дослідження будь-якими зацікавленими особами, а не тільки дослідниками-професіоналами, тобто відбувається своєрідна «демократизація» даної сфери. Проте, ряд авторів висловили заклопотаність з приводу можливих негативних наслідків зазначеного явища, перш за все, методологічних і етичних, а також небезпеки пересичення респондентів участю в опитуваннях.

Ще в 1994 році було поставлено питання про зменшення частки респондентів, які погодилися взяти участь в опитуванні (Schuldt and Totten 1994 [12]). Comley (1996) [13] відзначав, що в силу новизни використовуваної технології частка респондентів, які відгукнулися на e-mail-опитування, склала 45% в порівнянні з 16%, які відповіли на той же опитувальник, поширений за допомогою звичайної поштової розсилки.

До теперішнього часу метод онлайн-опитування довів свою спроможність і вже не сприймається з побоюванням дослідниками, як це було кілька років тому. Онлайн-опитування зайняв свою нішу поряд з традиційними методами поквартирного, телефонного, поштового та інших опитувань. Здебільшого визнання методу стало можливо завдяки повсюдної інтернетизації, що призвело до усунення недоліків пов'язаних з репрезентативністю вибірки, якість якої залежить від доступності Інтернету потенційної вибіркової сукупності. Таким чином, в найближчому майбутньому очікується розвиток методу, що дозволяє проводити все більш репрезентативні дослідження, так як зараз, не дивлячись на істотні плюси онлайн-методу, деколи складно закрити очі на якість отриманих даних. Як тільки Інтернет і інші супутні технології стануть доступними серед генеральної сукупності, буде набагато легше вирішити проблему репрезентативності [14].

З огляду на зростаючі витрати, пов'язані з традиційними методами опитування «обличчям-до-обличчя», популярність онлайн-панелей продовжить рости . Вже зараз вони досить поширені і утворили специфічний ринок, де у кожного є свої конкурентні переваги. Всі панелі постійно працюють над розширенням функціональності, що дозволяє справлятися з існуючими проблемами.

Розвиток онлайн-опитування сприяє розвитку самої методології опитування, так як онлайн платформа дозволяє фіксувати те, що раніше було недоступне за допомогою традиційних друкованих опитувальників або телефонних опитувань. Так, наприклад, завдяки онлайн-формату з'явилася можливість використовувати додаткові мультимедійні матеріали. Респонденту може бути попередньо надана для ознайомлення музична, графічна або відео інформація, а потім досліджено його думку щодо прочитаного, побаченого, почутого [15]. Більш того з'являється можливість аналізувати другорядні дані, збір яких раніше був неможливий. Це параданні (paradata) - дані про процес збирання даних (час початку і час завершення проходження опитування, час витрачений для заповнення певних блоків або окремих питань, геолокація і т. д.) . Все це веде до трансформації самої методології опитування, що в майбутньому може позначитися на більш якісних отриманих даних. Так, наприклад, за допомогою параданних можна буде вирішити проблему помилкових і довільних відповідей, які можна буде обчислювати за часом відповіді респондента або з якої-небудь іншої інформації про сам процес проходження опитування. Так само технічний розвиток методу може дозволити вирішити проблему кількаразового проходження анкетування. [16]

Таким чином, можна сказати, що головний напрямок розвитку даного методу є усунення існуючих недоліків, що в свою чергу призведе до розширення меж застосування онлайн-опитувань.

Отже, можна виділити такі переваги онлайн опитування:

– економія ресурсів (не тільки грошей, але часу і трудовитрат);

– великий обсяг вибірки;

– швидкість опитування (можливо опитати декілька тисяч чоловік за короткий термін);

– можливість оперативного реагування (наприклад, зміна інструментарію);

– широта охоплення (долання кордонів і відстаней, доступ до різних соціальних груп і спільнот);

– досяжність (можливість опитати тих, хто недоступний в реальному житті, наприклад, маргінальні групи);

– націленість (можлива побудова специфічної вибірки);

– релевантність (самостійність) комунікації, тобто нижчий рівень впливу інтерв'юера на респондента, можливість давати більш розгорнуті відповіді;

– високий рівень довіри (обумовлений анонімністю онлайн-середовища);

– широта охоплення предметний полів (можливість вивчати делікатні і закриті для публічного обговорення теми);

– організаційна гнучкість (респондент сам вибирає час і місце участі);

– сувора логіка проведення опитування (спеціальне програмне забезпечення дозволяє виключити традиційних помилок заповнення анкети);

– оперативний контроль за ходом заповнення анкети (наприклад, виявлення логічних протиріч у відповідях і їх виправлення).

– Але, в такому методі є і свої недоліки:

– відсутність репрезентативності (структура населення не збігається зі структурою користувачів);

– охоплення аудиторії може не відповідати шуканої аудиторії (наприклад, обмежуватися відвідувачами одного сайту);

– неодноразова участь в опитуванні (особливо при анонімному опитуванні);

– відсутність даних про генеральної сукупності (наприклад, про структуру аудиторії порталу або форуму);

– навмисне спотворення даних;

– можливість ворожих дій («злом» програмного забезпечення);

– комунікаційні проблеми (невірна інтерпретація питань анкети, помилки в переходах, заповненні таблиць та ін.);

1.1.2 Аналіз питання обробки даних

Обробка даних – це перетворення даних у зручну та бажану форму. Це перетворення або "обробка" здійснюється за допомогою визначеної послідовності операцій вручну або автоматично. Більша частина обробки здійснюється за допомогою комп’ютерів і, таким чином, робиться автоматично. Вихідні або "оброблені" дані можуть бути отримані в різних формах. Приклад цих форм включає зображення, графік, таблицю, векторний файл, аудіо, діаграми або будь-який інший бажаний формат. Отримана форма залежить від програмного забезпечення або методу обробки даних, що використовується. Після завершення роботи це називається автоматичною обробкою даних.

Обробка даних – це в основному синхронізація всіх даних, що вводяться в програмне забезпечення, щоб відфільтрувати найкориснішу інформацію з неї. Це дуже важливе завдання для будь-якої компанії, оскільки допомагає їм витягувати найбільш релевантний вміст для подальшого використання. Кожен важливий сектор, будь то банки, школи, коледжі чи великі компанії, майже всі потребують такої обробки даних. Ця обробка проводиться з метою збереження найдосконалішої інформації у своїх системах для подальшого використання. Ручна обробка вимагає багато часу і вимагає залучення занадто багато людей для цього. Це дійсно не є можливим завданням, коли у вас є дані масово. Сьогодні люди в галузі залежать від потужних та добре ефективних програмних засобів, які допоможуть обробити всі ці дані. Це допомагає їм досягти більшої точності та підвищити їх ефективність. При правильній обробці даних можна можна відсортувати все більше інформації. Це допомагає отримати більш чітке уявлення про матерію та краще розуміти її. Це може призвести до підвищення продуктивності та отримання прибутку для різних галузей бізнесу.

При належній обробці даних дослідники можуть писати наукові матеріали та використовувати їх у навчальних цілях. Також можна застосувати для оцінки економічних напрямків та факторів. У галузі охорони здоров’я оброблювані дані можуть бути використані для швидшого пошуку інформації та навіть для врятування

життя. Крім того, відомості про хвороби та записи методів лікування можуть зробити це менш трудомістким для пошуку рішень та допомогти зменшити страждання пацієнтів.

В даний час збирається все більше даних для академічних, наукових досліджень, приватного та особистого використання, інституціонального та комерційного використання. Ці зібрані дані потрібно зберігати, сортувати, фільтрувати, аналізувати та представляти і навіть вимагати передачі даних для будь-якого використання. Цей процес може бути простим або складним залежно від масштабу, в якому здійснюється збір даних, і складності результатів, які потрібно отримати. Час, витрачений на отримання бажаного результату, залежить від операцій, які необхідно виконати зібраними даними, та від характеру вихідного файлу, необхідного для отримання. Ця проблема стає головною при роботі з дуже великим обсягом даних. Наприклад, дані, зібрані багатонаціональними компаніями. Вони збирають дані про своїх користувачів, продажі, виробництво тощо. Такі служби та компанії, що займаються персональною інформацією та іншою конфіденційною інформацією, повинні бути обережними щодо захисту даних.

Потреба в обробці стає все більш критичною в таких випадках. У таких випадках грають дані та управління даними, без яких оптимальних результатів неможливо отримати. Кожен етап, починаючи від збору даних до представлення, має прямий вплив на вихід та корисність оброблюваних даних.

Обробка даних для їх упорядкування за типом та інформацією може заощадити багато місця, зайнятого даними, які не організовані та зберігаються випадково.

Дані в будь-якій формі та будь-якого типу потребують опрацювання більшу частину часу. Ці дані можна класифікувати як особисту інформацію, фінансові операції, податкові кредити, банківські реквізити, обчислювальні дані, зображення та просто майже все, що ви можете придумати. Кількість необхідної обробки буде залежати від спеціалізованої обробки, якої вимагають дані. Згодом це буде залежати від потрібного вам результату. Зі збільшенням попиту та потреби в таких послугах з'явився конкурентний ринок послуг передачі даних.

1.2 Опис та аналіз сучасного стану проблеми

Багато компаній та організацій стикаються з труднощами при спробі дізнатися думку працівників або споживачів (клієнтів) про продукти або послуги, про колектив

або внутрішню організацію, про умови праці або якість обладнання. Респонденти наполегливо мовчать, а у відповідь на пряме прохання пишуть стандартну відписку виду «все добре».

Але, якщо компанія чи організація досить велика, процес опитування та обробки отриманої інформації перетворюється в надзвичайно складну проблему.

Керівництво компаній, зазвичай, використовує опитування через електронну пошту. Але чи завжди це зручно? Якщо в компанії працює більше 300 працівників, або середня кількість клієнтів в день перевищує 500 чоловік, організатор опитування отримає безліч повідомлень, які перетворюють його почтову скриньку на великий смітник, в якому буде майже неможливо проаналізувати всю інформацію. Таким способом легко опитати клієнтів, які користуються послугами, але, в такому випадку, гостро постає питання анонімності.

Якщо питання анонімності переходить на перший план, то на допомогу приходять паперові анкети. Але, процес їх обробки навіть більш клопіткий, ніж обробки електронних повідомлень.

Щодня дані генеруються, збираються у величезній кількості, але багато разів вони залишаються невикористаними без отримання корисної та змістовної інформації. Ці відомості є життєво важливими в процесі прийняття стратегічних і оперативних рішень, таких як маркетинг, залучення клієнтів, брендинг тощо. Дані, що генеруються різними каналами, такими як маркетинг, дистрибуція, залучення клієнтів, соціальні канали та веб-контент у різних структурованих, а також неструктурованих формах та доступними у багатьох системах. Ці неструктуровані дані потрібно перетворити на щось трохи корисніше. Це вимагає пошуку підходів до перетворення вільної форми неструктурованих даних до форми структурованих або напівструктурованих даних та проаналізувати їх для отримання змістовної відомості щодо вирішення проблеми або допомоги приймати бізнес-рішення.

Тому оптимально буде використовувати програмні системи, які допоможуть вирішити ці проблеми.

Але, чи завжди можна довіряти стороннім системам? Досить часто клієнтів та працівників просять передати компанії персональні дані, наприклад номер телефону, місце проживання, сімейний стан і т. д. Подібна інформація може бути передана конкурентам та навіть шахраям. Так, під удар може потрапити як бізнес, так і клієнт.

1.3 Аналіз існуючих систем для збору даних

Існує досить багато систем, які можуть допомогти у вирішенні проблеми збору даних. Розглянемо основні з них:

– Surveymonkey

Функціонал:

1. більше 15 типів питань;
2. призначений для користувача логотип і брендування;
3. призначені для користувача звіти;
4. результати в реальному часі;
5. інтеграція з MailChimp і іншими сервісами;
6. готові шаблони для опитувань;
7. рекомендації та навчальні матеріали;
8. можливість додати опитування на сайт, розсилки по e-mail, в соціальні мережі.

Головним недоліком Surveymonkey є ціна [18].

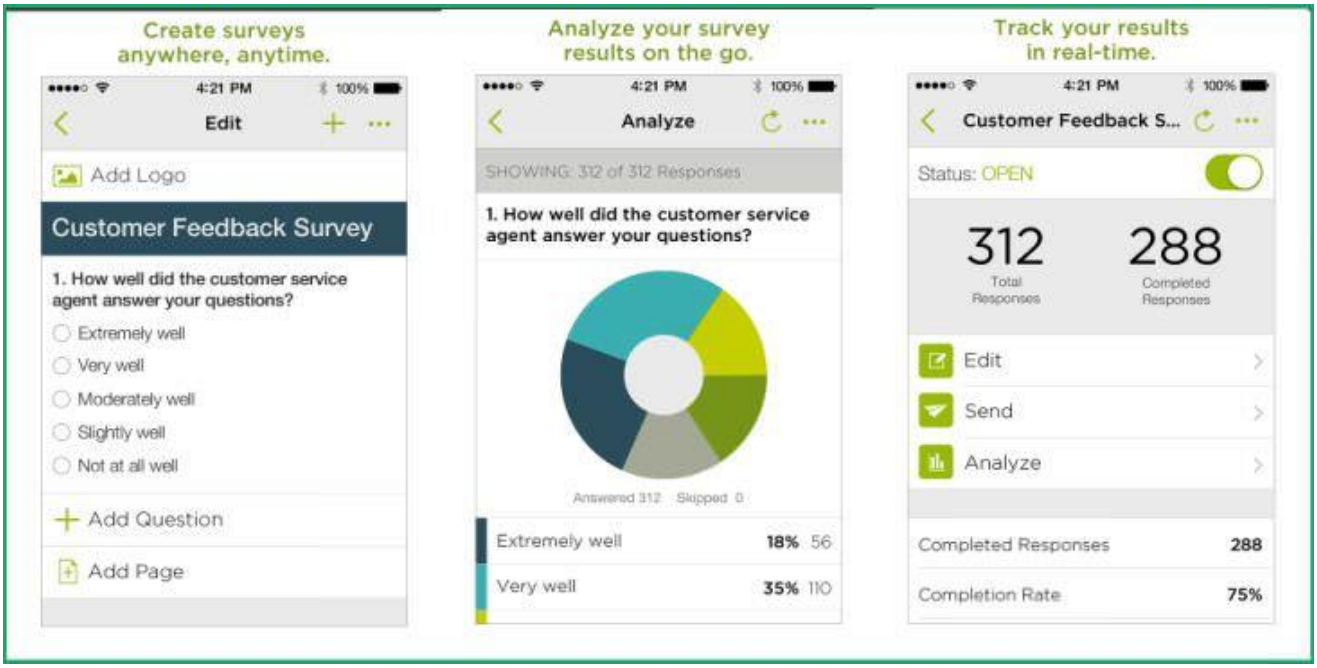

Рисунок 1.3.1 – Інтерфейс Surveymonkey

– Google Forms

Функціонал:

1. безкоштовний;
2. 9 типів питань;
3. можливість командної роботи;
4. логічні переходи між питаннями;
5. завантаження зображень і відео в питання;
6. можливість вставки на сайт або розсилки;
7. перегляд відповідей в таблицях Google;
8. відповіді прямо в формі;
9. швидкий перегляд відповідей;
10. можливість побудови діаграм.

Головними недоліками є низький рівень налаштування та проблема зберігання важливої персональної інформації [19].

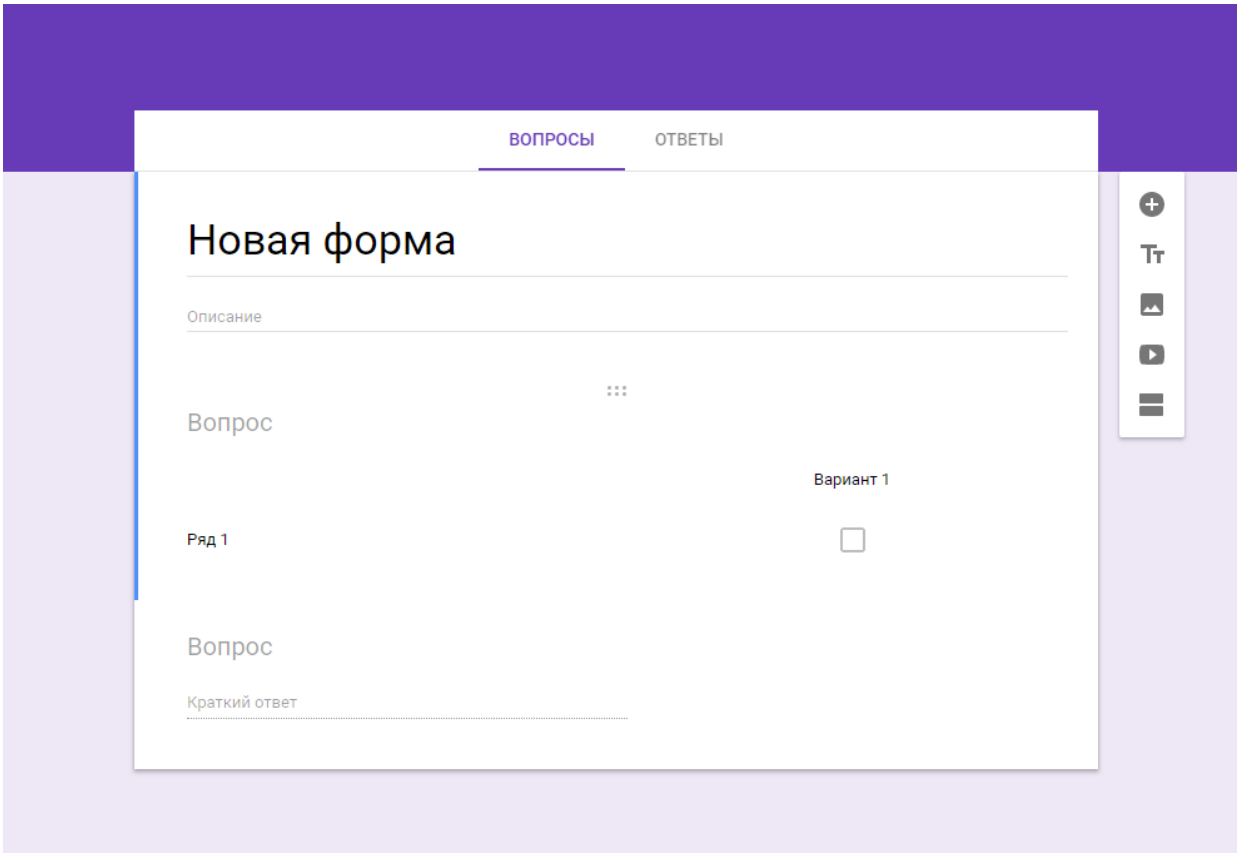

Рисунок 1.3.2 – Інтерфейс Google Forms

– Typeform

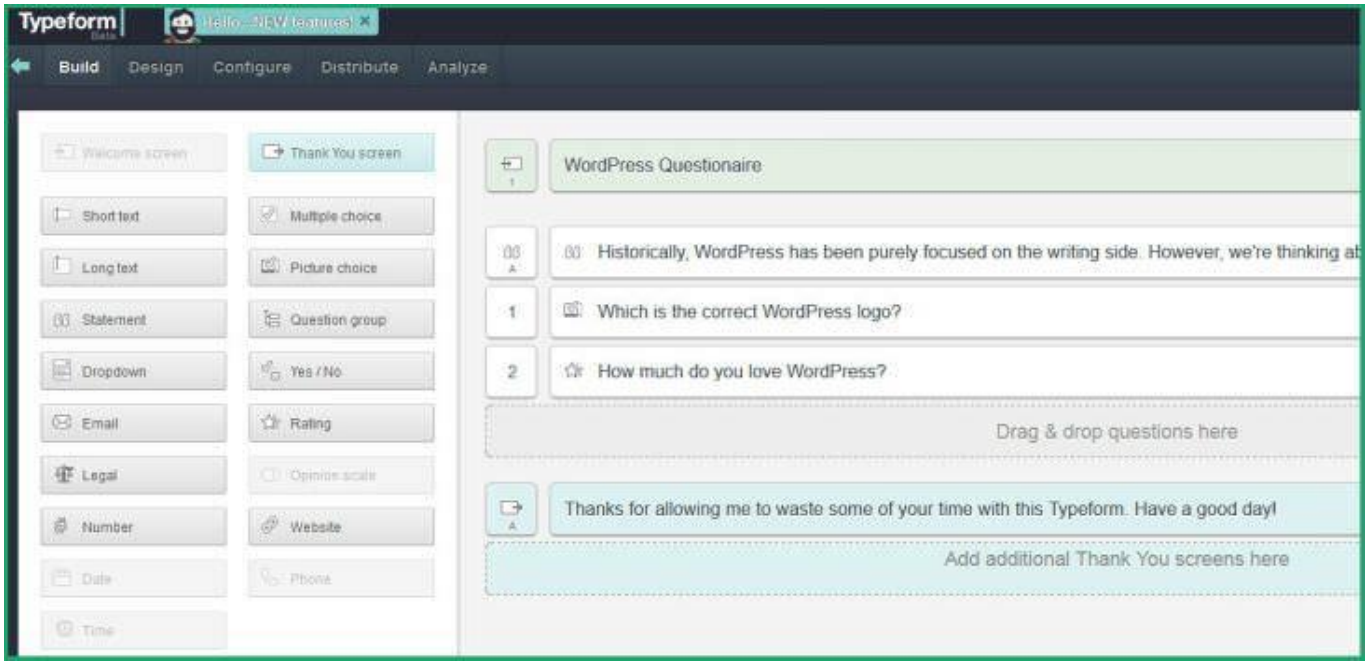

Рисунок 1.3.3 – Інтерфейс Typeform

Функціонал:

1. зручний і функціональний конструктор в режимі drag & build;
2. логічні переходи між питаннями;
3. можливість додавання відео та зображень;
4. калькулятор;
5. індивідуальний дизайн опитувань;
6. повідомлення;
7. можливість вставки опитування на сайт або розсилки по e-mail;
8. красиві і зрозумілі звіти за результатами опитувань;
9. експорт результатів.

Головними недоліками є ціна та проблема зберігання важливої персональної інформації [20].

– Anketolog

Функціонал:

1. створення анкет;
2. створення питань в анкеті;
3. створення відповідей на кожну анкету;
4. логічні розгалуження;
5. шаблони анкет;
6. налаштування колірних схем;
7. вбудовування відео в анкеті;
8. власна посилання на анкету;
9. результати;
10. експорт в PDF, Word, Excel.

Головними недоліками є:

1. 10 ГБ місця для файлів;
2. ціна;
3. не підтримує доступ по протоколу HTTPS;
4. не підтримує розповсюдження анкет через електронну пошту;
5. не підтримує повідомлення респондента про отримання нової анкети.

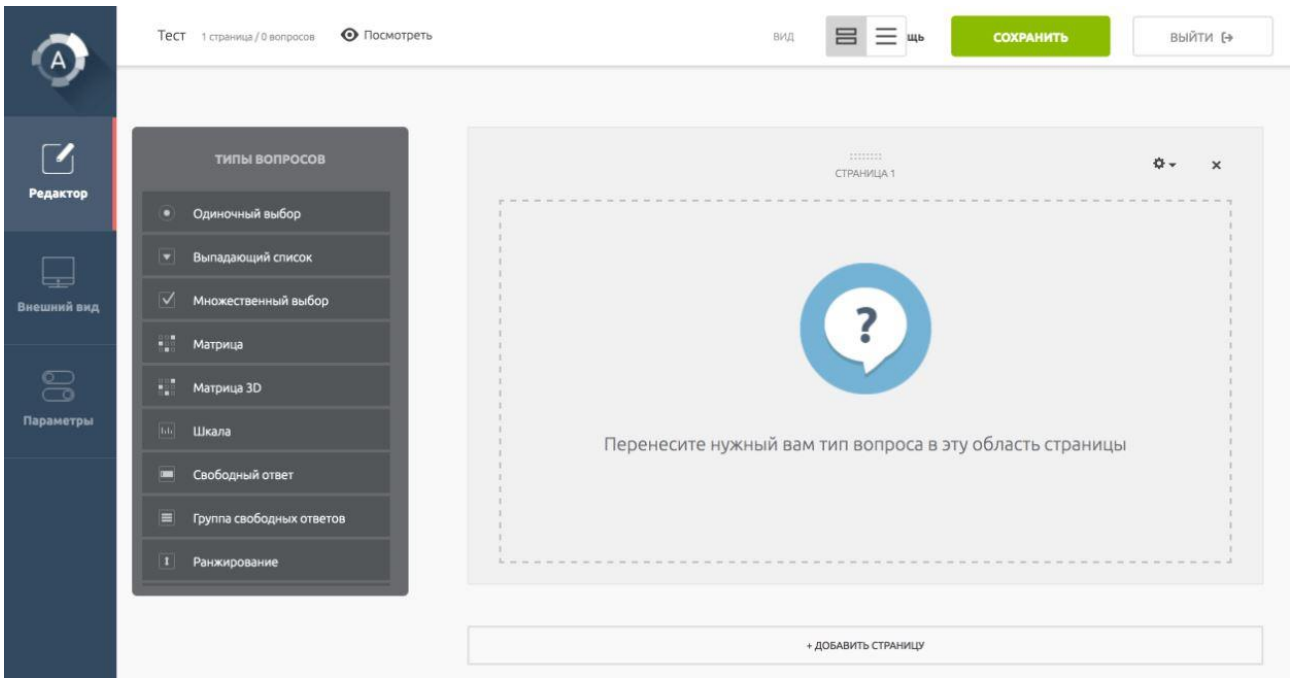

Рисунок 1.3.4 – Інтерфейс Anketolog

– Simpoll

Основні переваги:

1. багатомовність;
2. зручний інтерфейс конструктора;
3. керування логікою анкет;
4. налаштування колірних схем;

Основними недоліками є:

1. ціна;
2. в безкоштовній версії можна створити максимум 3 анкети по 10 питань;
3. в безкоштовній версії не підтримується основна частина функціоналу.

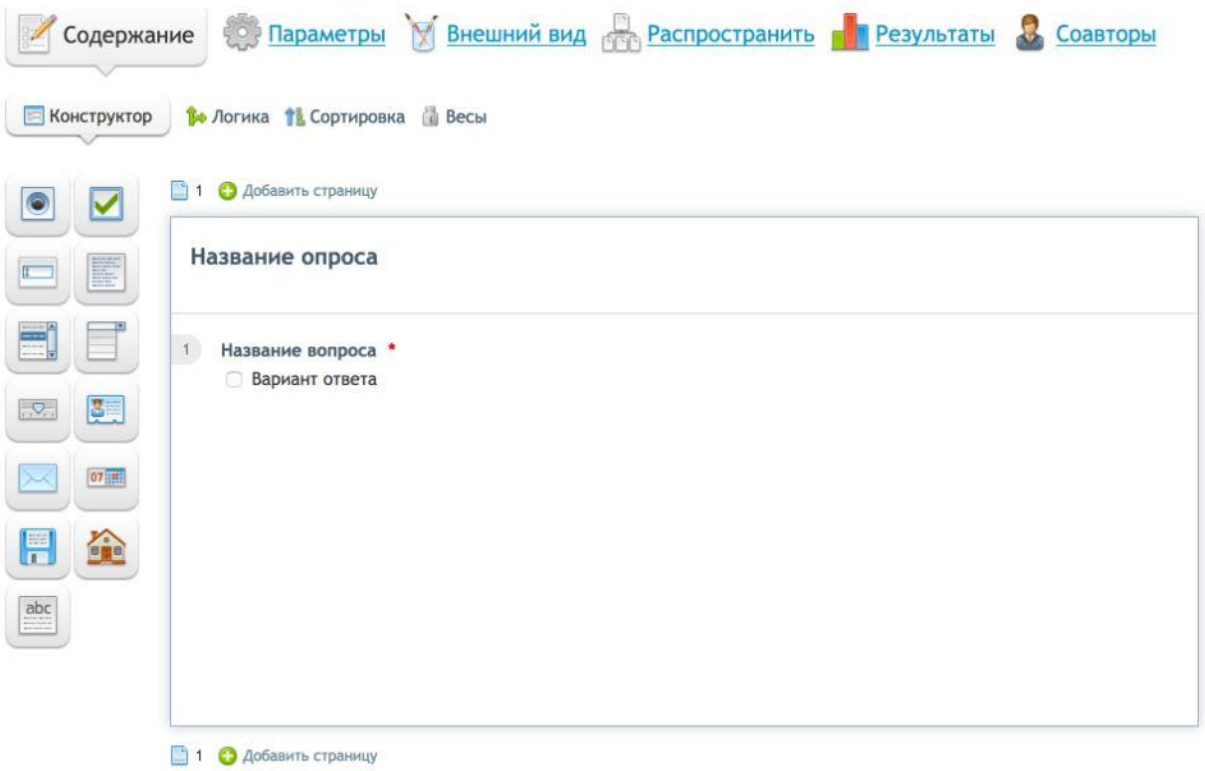

Рисунок 1.3.5 – Інтерфейс Simpoll

– Survio

Основними перевагами є:

1. необмежене число опитувань;

2. звіти в PDF, DOC, XLS, CSV, XML, HTML;
3. 100+ шаблонів анкет для швидкої їх публікації;
4. логіка анкети;
5. захист паролем + IP;
6. вставка свого логотипу в анкетах і звітах;
7. запрошення листами по E-mail;
8. обмін результатами;
9. таблиці та графіки.

Головними недоліками є:

1. ціна;
2. не підтримує доступ по протоколу HTTPS;

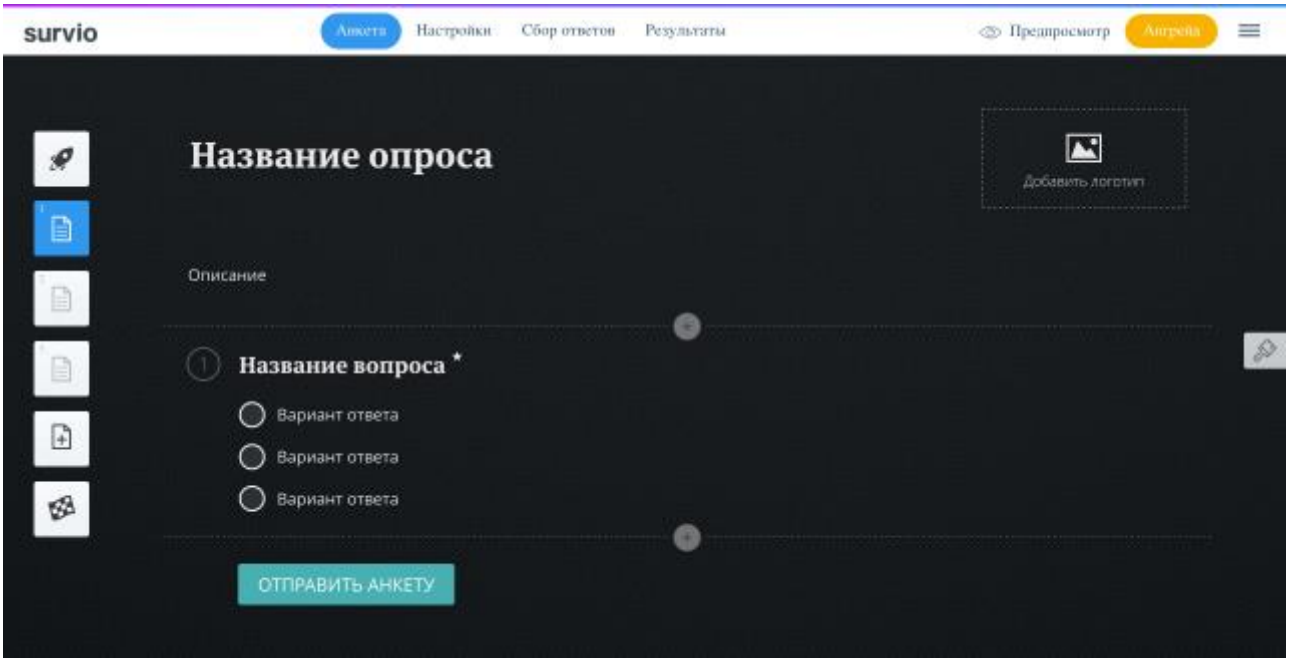

Рисунок 1.3.6 – Інтерфейс Survio

Таблиця 1.1 – Порівняльна таблиця аналгів

	Forms Generator	Surveymonkey	Google Forms	Typeform	Anketolog	Simpoll	Survio
Ціна	+	-	+	-	-	-	-
Налаштування	+-	+	+-	+-	+	+	+

Повідомлення	+	+	-	+	-	-	+
Безпека	+	+-	+-	+-	+-	+-	+-
Аналіз даних	+	-	-	-	-	-	-

1.4 Аналіз існуючих інструментів Big Data

Традиційні засоби обробки даних (наприклад, Excel, SPSS тощо) не в змозі збільшити масштаби зростаючих наборів даних. Наприклад, за допомогою Microsoft Excel 2013 аналітики не можуть проводити аналіз більш ніж на 1 мільйон рядків. Отже, інструмент, який може розширювати масштаб, слід використовувати для боротьби зі зростаючими наборами даних. Платформи соціальних мереж, такі як Facebook, Twitter тощо, генерують величезні обсяги цінних соціальних даних на день з дуже високою швидкістю. Ці дані потрібно обробляти в режимі реального часу, які можна використовувати для прогнозування результатів виборів, поведінки на фондовому ринку тощо. Для цього нам потрібні інструменти, які можуть проводити аналіз потокової передачі даних. Основні інструменти для обробки різних характеристик великих даних узагальнені в таблиці I.

Реляційні бази даних MPP (Massively Parallel Processing), такі як Greenplum, Vertica тощо, мають можливість зберігати та керувати петабайтами даних, де дані розподіляються на декілька вузлів, кожен вузол має процесори/пам'ять для локальної обробки даних (це архітектура спільного використання, тобто обмін на рівні диска). Але бази даних MPP мають верхню межу ємності пам’яті, а також ті самі обмеження для обробки даних, що й асоційовані з SQL.

Напівструктуровані дані - це структуровані дані, що зберігаються у формі, відмінній від таблиць (наприклад, XML, JSON тощо), а система управління базами даних, що забезпечує механізм зберігання та пошуку таких даних, називається сховищем даних NoSQL (NoSQL розшифровується як Не тільки SQL, що вказує на те, що вони також можуть підтримувати SQL-подібні мови запитів). Прикладами сховищ даних NoSQL є Cassandra, HBase, MongoDB тощо. У сховищах даних NoSQL немає фіксованої жорсткої схеми, як RDBMS для введення даних, отже, вони можуть обробляти різні дані, що надходять з різних джерел. BigTable [21] – це розподілена система зберігання даних, призначена для управління великими обсягами напівструктурованих даних.

Таблиця 1.2 Інструменти Big Data

Характеристика	Інструмент	Примітка
Обсяг	MPP databases, Hadoop	розподілена обробка
Різноманітність	NoSQL databases, Hadoop	сховище даних без схем
Швидкість	In-memory databases, Spark	запитувати дані всередині оперативної пам'яті замість диска
Обробка потоків	Storm, S4	Обробка в режимі реального часу замість пакетної обробки

Файлова система Google (GFS) [22] - це розподілена файлова система, призначена для забезпечення доступу до даних за допомогою великих кластерів commodity hardware.

Map-Reduce [23] - модель програмування для розподіленої та паралельної обробки. Модель складається з двох функцій: функція «Map» розбиває вхідні дані та розподіляє їх по декількох центральних каналах для паралельної обробки даних та функція «Reduce», яка збирає результати з машин та повторно розв'язує ці результати до кінцевого сумарного результату. Програмування Map-Reduce може бути виконано на Java, Python, Ruby тощо.

Apache Hadoop - це інструмент з відкритим кодом для розподіленого зберігання та розподіленої обробки. Hadoop використовує HDFS (розподілену файлову систему Hadoop), яка базується на GFS для розподіленого зберігання напівструктурованих даних, і використовує Map-Reduce для розподіленої обробки.

Amazon Elastic MapReduce (EMR) - веб-сервіс, який використовує Hadoop для обробки великих обсягів даних на хмарі Amazon EC2 (Elastic Compute Cloud).

Існують різні інструменти на основі Hadoop, які роблять обробку Big Data більш зручною та ефективною. Apache Flume - це інструмент для збору, агрегації та переміщення даних з різних джерел в Hadoop. Apache HBase - це open-source distributed сховище даних NoSQL, модельоване на BigTable, і працює над HDFS. HBase дозволяє зберігати дані у вигляді таблиць (але дані, що зберігаються, не потребують заздалегідь визначеної фіксованої схеми, як у RDBMS). HBase допомагає

Hadoop долати проблеми у випадковому читанні та записі. HBase не забазпечує мову запитів, подібних SQL, і для обробки даних у таблицях HBase ми повинні використовувати програми Map-Reduce. Apache Hive - це система зберігання даних, побудована на базі Hadoop, яка дозволяє користувачам зберігати дані в таблицях і писати запити в HiveQL (мова запитів Hive) для отримання даних замість складної програми Map-Reduce. Hive перетворює запити HiveQL в ряд завдань map-reduce. Але HiveQL має обмежені можливості, і його не можна використовувати для складних операцій. Cloudera, MapR, Horton-works, IBM Infosphere BigInsights - це деякі дистрибутиви на базі Hadoop, що забезпечують HDFS, Map-Reduce, Pig, Hive, HBase, Sqoop, Flume, Hue, Impala, Oozie, ZooKeeper, Sentry тощо в одному пакеті.

Наразі обговорювані інструменти стосуються лише "обсягу", і нам потрібно шукати інструменти, які також можуть обробляти потоки даних. Бази даних у пам'яті, такі як SAP HANA, Altibase тощо, набувають популярності у додатках, де час відгуку є критичним, але майже всі вони є RDBMS. Apache Storm та Apache S4 (прості масштабовані потокові системи) розподіляються в обчисленні в режимі реального часу для оброблення швидких великих потоків даних. Apache Spark - це система кластерних обчислень, яка забезпечує продуктивність у 100 разів швидше, ніж Map-Reduce, завдяки кешуванню даних. Apache Spark також може обробляти потоки даних через бібліотеку Spark Streaming. Apache Hadoop YARN - це підпроект Hadoop, який дозволяє Hadoop вийти за межі пакетної обробки та підтримувати більш широку обробку даних. YARN (Yet Another Resource Negotiator) є частиною Hadoop NextGen, і дозволяє Hadoop надавати можливості управління ресурсами для систем, таких як Apache Spark, тощо. Apache Samza є структурою потокової обробки потоків на основі YARN. Amazon Kinesis - хмарний сервіс для обробки розподілених потоків даних.

1.5 Постановка задачі

Враховуючи мету даної кваліфікаційної роботи, в процесі її написання необхідно:

- провести дослідження можливостей алгоритмів пошуку ассоціативних правил для обробки неструктурованих даних;
- розробити метод обробки неструктурованих даних на основі генерації асоціативних правил та технологій Big Data та Data Mining;

- визначити вимоги до прототипу системи збору та обробки неструктурованих даних за на основі нового методу генерації асоціативних правил та за допомогою технологій Big Data та Data Mining.

Головними вимогами до системи є високий рівень налаштування, легкість у використанні, швидкість, безпечність, адаптованість під мобільні пристрої. Також повинен підтримувати можливість бути доступним тільки з приватних мереж та не порушувати правил компаній.

Розроблюваний програмний засіб повинен виконувати наступні завдання:

1. додавання та редагування елементів форм;
2. додавання валідації на елементи форм;
3. створення та редагування форм;
4. створення та редагування колекцій форм;
5. запрошення респондентів по електронній адресі та за посиланням;
6. можливість пройти колекцію форм;
7. перегляд результатів;
8. обробка результатів за допомогою нового методу пошуку асоціативних правил та технологій Big Data та Data Mining;
9. експорт відповідей.

- спроектувати прототип системи на основі вимог, визначених на попередньому етапі;

На даному етапі потрібно визначити ролі, описати варіанти використання та Use-Case діаграмами. Також потрібно визначитись з мовою програмування, СУБД, архітектурою додатка. Спроектувати інтерфейс користувача.

- створити прототип на основі створеної документації з проектування;

При розробці потрібно враховувати SOLID принципи розробки програмного забезпечення, а також дотримуватися методологій DRY та YAGNI.

- провести оцінювання ефективності роботи прототипу системи, в якій буде використано новий метод пошуку асоціативних правил для неструктурованих даних.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ПОШУКУ АСОЦІАТИВНИХ ПРАВИЛ

2.1 Алгоритм AIS

Алгоритм AIS робить кілька проходів по всій базі даних. Під час кожного проходу він сканує всі транзакції. У першому проході він рахує підтримку окремих елементів і визначає, які з них є великими або частими в базі даних. Великі набори елементів кожного проходу розширюються, щоб генерувати набори елементів-кандидатів. Після сканування транзакції визначаються загальні набори елементів між великими наборами предметів попереднього проходу та предметами цієї транзакції. Алгоритм AIS був першим опублікованим алгоритмом, розробленим для генерації всіх великих наборів елементів у базі даних транзакцій. Основна увага була зосереджена на розширенні баз даних з необхідною функціональністю для обробки запитів підтримки рішення. Цей алгоритм був спрямований на виявлення якісних правил.[35][36][37]

Переваги:

- Алгоритм був використаний, щоб визначити, чи існує зв'язок між відділами в поведінці покупця клієнта.

Недоліки:

- Основна проблема алгоритму AIS полягає в тому, що він генерує занадто багато кандидатів, які згодом виявляються малими
- Структури даних, необхідні для підтримки великих і наборів наборів елементів, не були визначені

Алгоритм роботи:

- Набори кандидатів створюються та підраховуються під час сканування бази даних.
- Для кожної транзакції визначається, який з великих наборів предметів попереднього пропуску міститься в цій транзакції.
- Нові набори наборів елементів генеруються шляхом розширення цих великих наборів елементів на інші елементи цієї транзакції.

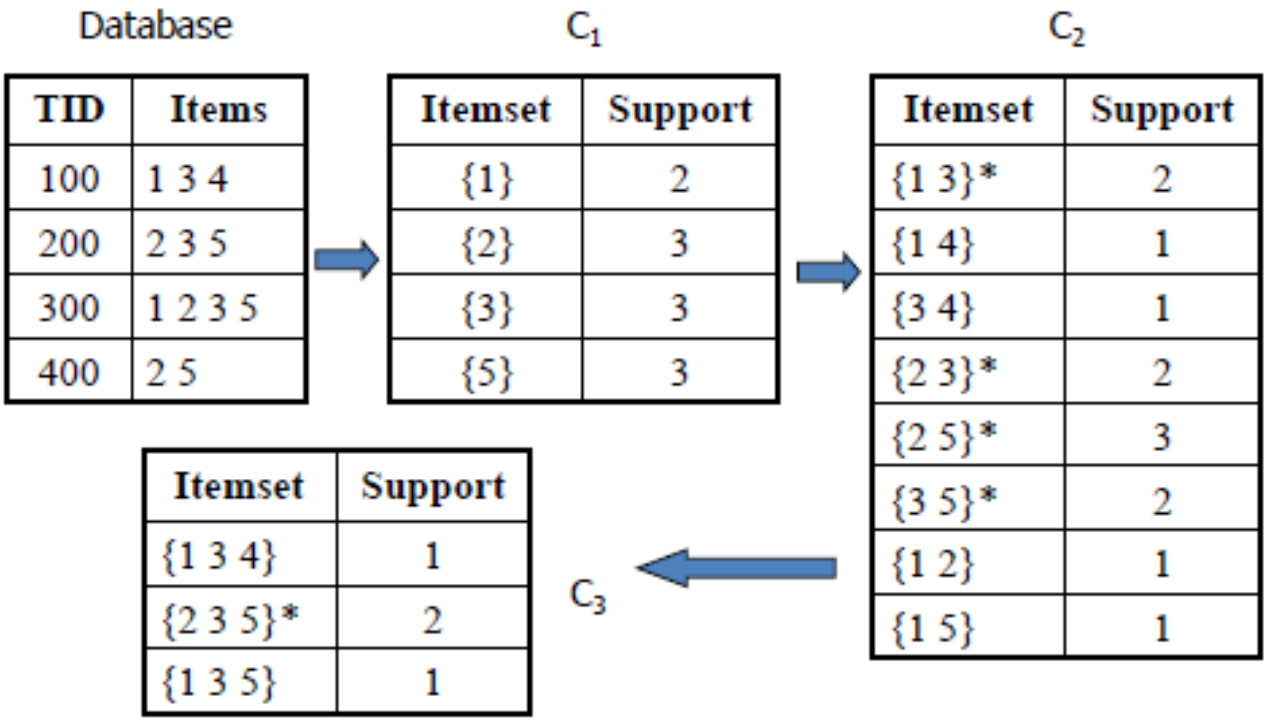

Рисунок 2.1 – Алгоритм AIS

2.2 Алгоритм SETM

Подібно до алгоритму AIS, алгоритм SETM робить кілька проходів по базі даних. У першому проході він рахує підтримку окремих елементів і визначає, які з них є великими або частими в базі даних. Потім він генерує набір елементів-кандидатів, розширюючи великі набори елементів попереднього проходу. Крім того, SETM запам’ятовує TID про генеруючі транзакції з набором елементів набору. Реляційна операція злиття-з'єднання може використовуватися для генерації наборів елементів[37][38][39].

Переваги:

- Генеруючи набори кандидатів, алгоритм SETM зберігає копію наборів елементів-кандидата разом з TID породжувальної транзакції послідовно.

Недоліки:

- Оскільки для кожного набору елементів-кандидата є пов'язаний з ним TID, для зберігання великої кількості TID потрібно більше місця.
- SETM не ефективний і немає інформації про результати його запуску з реляційними СУБД.

Алгоритм роботи:

- Набори елементів кандидата створюються під час руху під час сканування бази даних, але підраховуються в кінці пропуску.
- Нові набори елементів-кандидатів генеруються так само, як і в алгоритмі AIS, але TID транзакції, що генерує, зберігається з набором елементів-кандидата в послідовній структурі [41].
- В кінці пропуску визначається кількість підтримки наборів елементів кандидатів шляхом агрегування цієї послідовної структури.

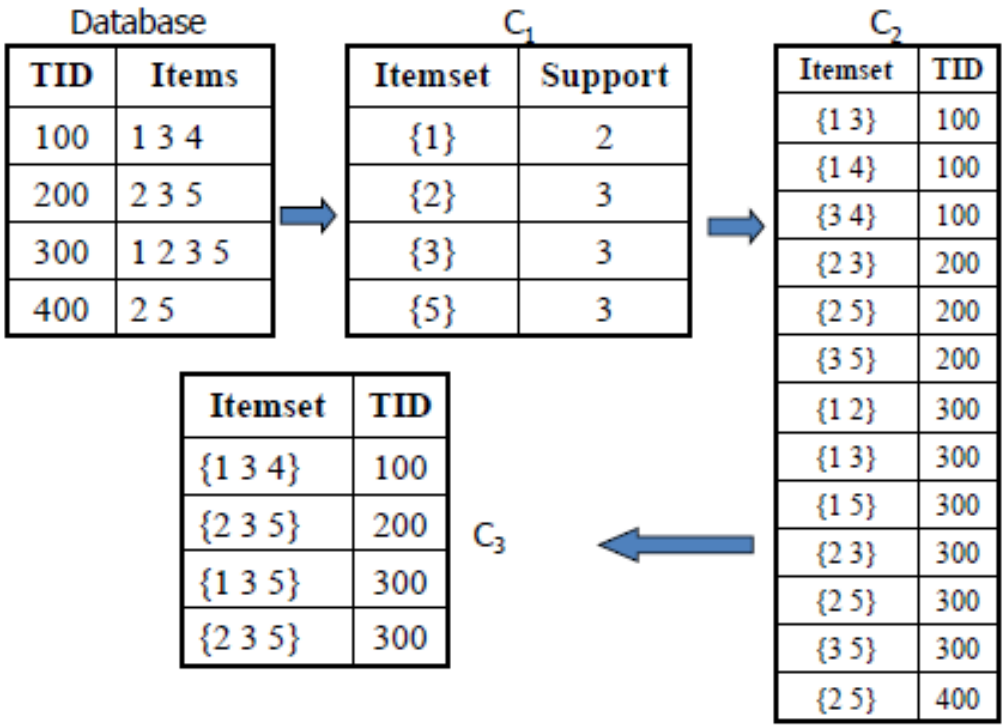

Рисунок 2.2 – Алгоритм SETM

2.3 Алгоритм Apriori

Це на сьогоднішній день найбільш відомий алгоритм правил асоціації. Принципові відмінності цього алгоритму від алгоритмів AIS та SETM полягають у формуванні наборів наборів елементів та виборі наборів елементів для підрахунку. Apriori генерує набір елементів-кандидатів, приєднуючись до великих наборів елементів попереднього проходу та видаляючи ті підмножини, які є малими в попередньому проході, не враховуючи транзакцій у базі даних. Враховуючи лише

великі набори предметів попереднього пропуску, кількість великих наборів кандидатів значно скорочується [42][43][44][45][46].

Алгоритм роботи:

- Кандидатські набори елементів генеруються за допомогою лише великих наборів елементів попереднього пропуску без урахування транзакцій у базі даних.
- Великий набір предметів попереднього проходу поєднується з собою, щоб генерувати всі набори елементів, розмір яких вище на 1.
- Кожен згенерований набір елементів, який має велике підмножина, видаляється. Решта наборів елементів – це кандидати [47].

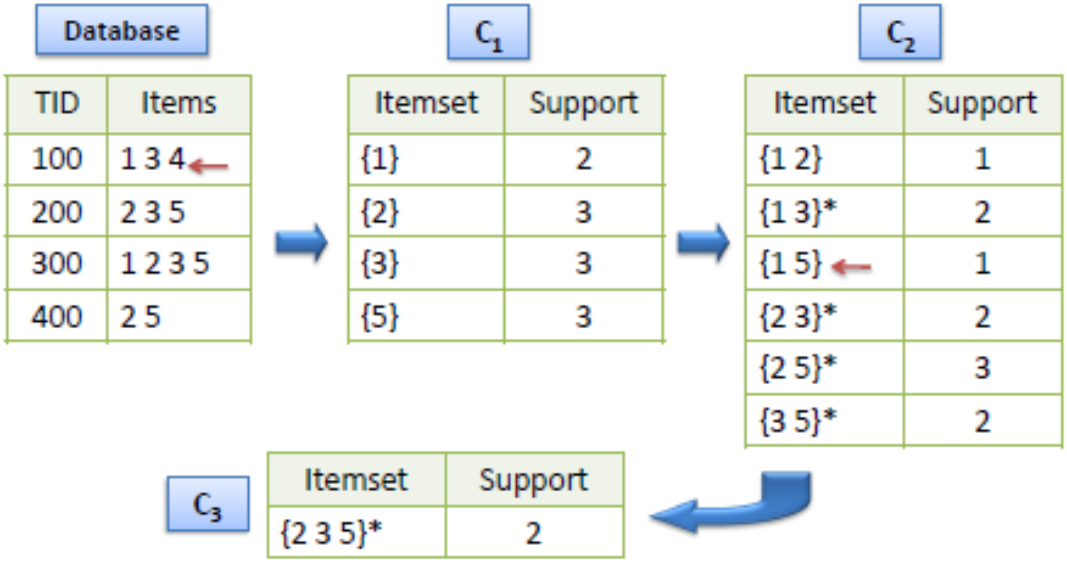

Рисунок 2.3 – Алгоритм Apriori

3 ОПИС СТРУКТУРИ ТА РОБОТИ СИСТЕМИ

3.1 Опис структури блока збору інформації

Система, що розробляється являє собою сайт, який дозволяє збирати певні дані користувачів. Збір даних проводиться в формі опитування (заповнення форми на сайті).

Головними поняттями в системі, що розробляється є форми та колекції форм.

Форма – це набір елементів управління (текстові поля, поля для вводу дати і т. д.), які згруповані по певному логічному ознаку (наприклад, прізвище те ім’я).

Якщо розглядати форму, як певну математичку структуру, то можна уявити її як множину (формула 3.1).

$$F = \{C_1, C_2, \dots, C_n\}, \quad (3.1)$$

де F – форма,
C – елементи управління,
n – кількість елементів управління на форму, де

$$C \in \{textarea, textbox, checkbox, radio, dropdown, date, file\}$$

В свою чергу, колекція форм – це набір форм, згрупованих по певному логічному або змістовному ознаку.

Колекцію форм описує формула 3.2.

$$FC = \{F_1, F_2, \dots, F_m\}, \quad (3.2)$$

де FC – колекція форм,
F – форми, з яких складається колекція форм, .
m – кількість форм в даній колекції.

3.2 Опис структури блока обробки інформації

Алгоритм Apriori, є одним з найпопулярніших основних алгоритмів для створення необхідних булевих асоціаціативних правил для видобутку частих наборів елементів. Алгоритм Apriori видобуває часті набори елементів у циклічному ієрархічному порядку пошуку. У цьому алгоритмі є одна відома властивість – весь непустий дочірній набір частих наборів предметів повинен бути частим [40].

Алгоритм Апріорі використовує ітеративний метод поетапного пошуку, основна ідея якого полягає в тому, що набір елементів-елементів k створюється з побудованих частих наборів елементів $k-1$. По-перше, генеруйте часті набори елементів P_1, а потім використовуйте P_1 для генерації P_2 (а саме часті два набори). P_2 використовується для генерації P_3 тощо, поки не з’являться нові часті набори елементів k [48][49][50]. Алгоритм управління потоком показаний на рисунку 3.1.

Метод для генерування частих кандидатських наборів елементів складається з двох основних етапів:

(1) шляхом підключення будь-яких двох частих наборів елементів $k-1$ у частих наборах предметів S_{k-1} для створення кандидатських частих наборів елементів S_k.

(2) використовувати стратегії обрізки на частих наборах елементів кандидатів, породжених (1), щоб вирізати нечасті набори елементів, а потім пройти S_k і визначити, чи часто кожен елемент у S_k є властивістю Apriori.

Алгоритм Apriori був використаний як класичний алгоритм виведення асоціативних правил [51][52]. Однак, з опису алгоритму не важко зрозуміти, що алгоритм Apriori існує внаслідок таких недоліків [53][54]:

- розмір частих наборів елементів, що генерується з частих наборів елементів $k-1$, шляхом автоматичного з'єднання будь-якого з двох елементів, величезний , що є експоненціальним зростанням [55][56]. Складність програмного простору та часова складність у формуванні частих наборів кандидатів є величезною проблемою [57][58];
- перевірка наборів частих кандидатів k часто потребує частого сканування всієї таблиці, що вимагає великих витрат часу та завжди спричиняє вузькі місця вводу/виводу [59][60].

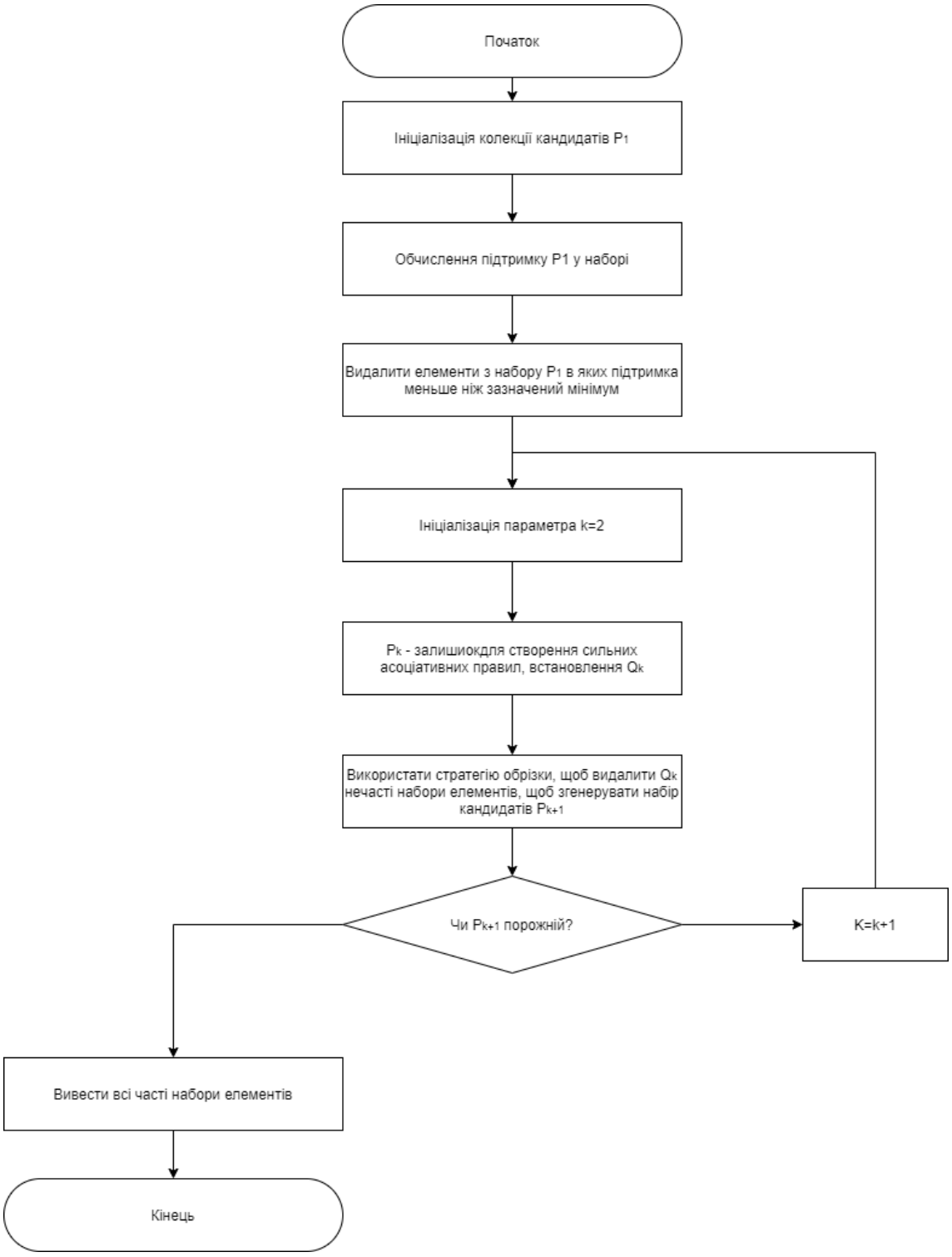

Рисунок 3.1 – Алгоритм Apriori

3.3 Загальний опис роботи системи

Програмний засіб умовно поділити на 3 частини: створення форм, створення колекції форм, проходження колекції форм та аналіз отриманих даних.

При створенні форми потрібно перш за все додати ім’я форми та додати всі необхідні елементи управління. При необхідності задати додаткові параметри, такі як валідація, та значення за замовчуванням. В кінці зберегти форму.

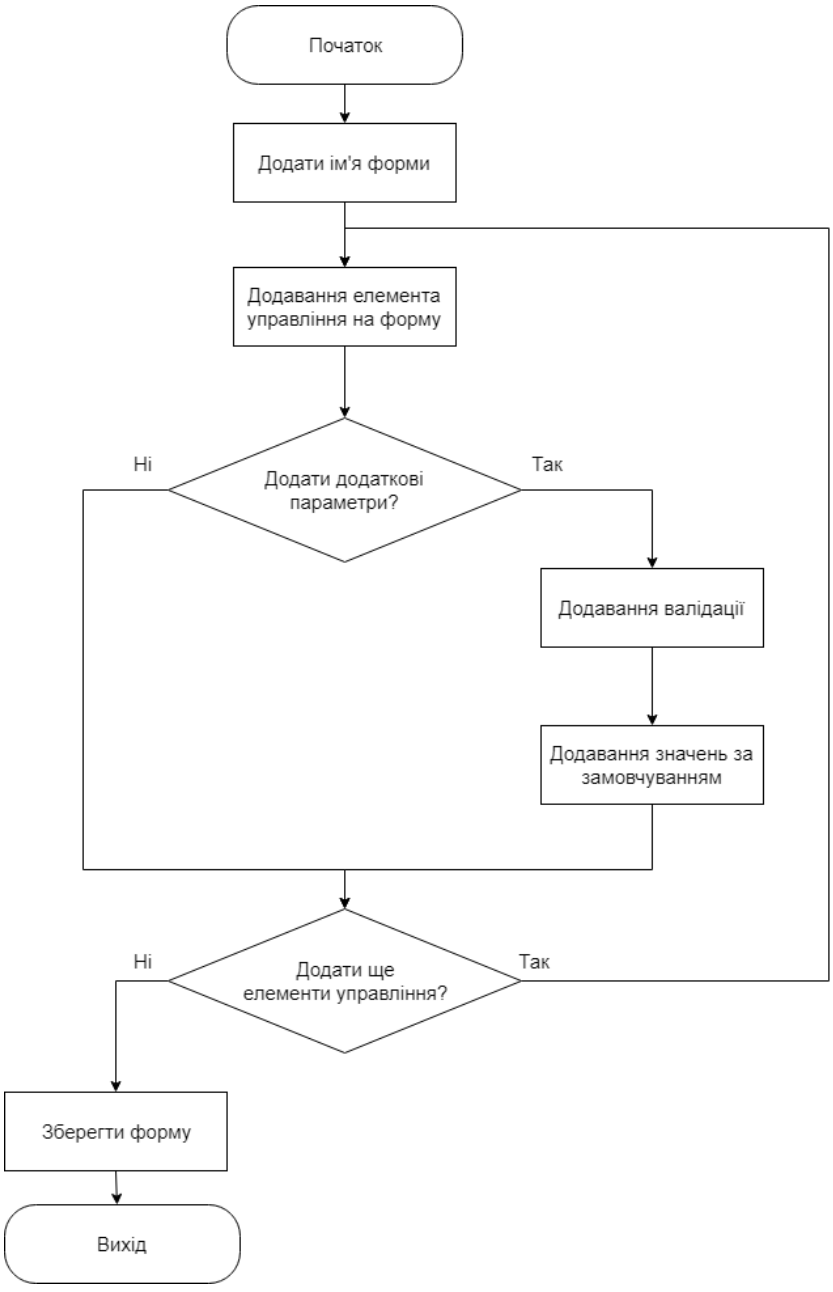

Рисунок 3.1 – Алгоритм створення форми

При створенні колекції форм, нам необхідно перш за все ввести ім’я колекції форм. Потім знайти необхідні форми з параметрами пошуку або ні та додати їх до форми. Зберегти колекцію форм.

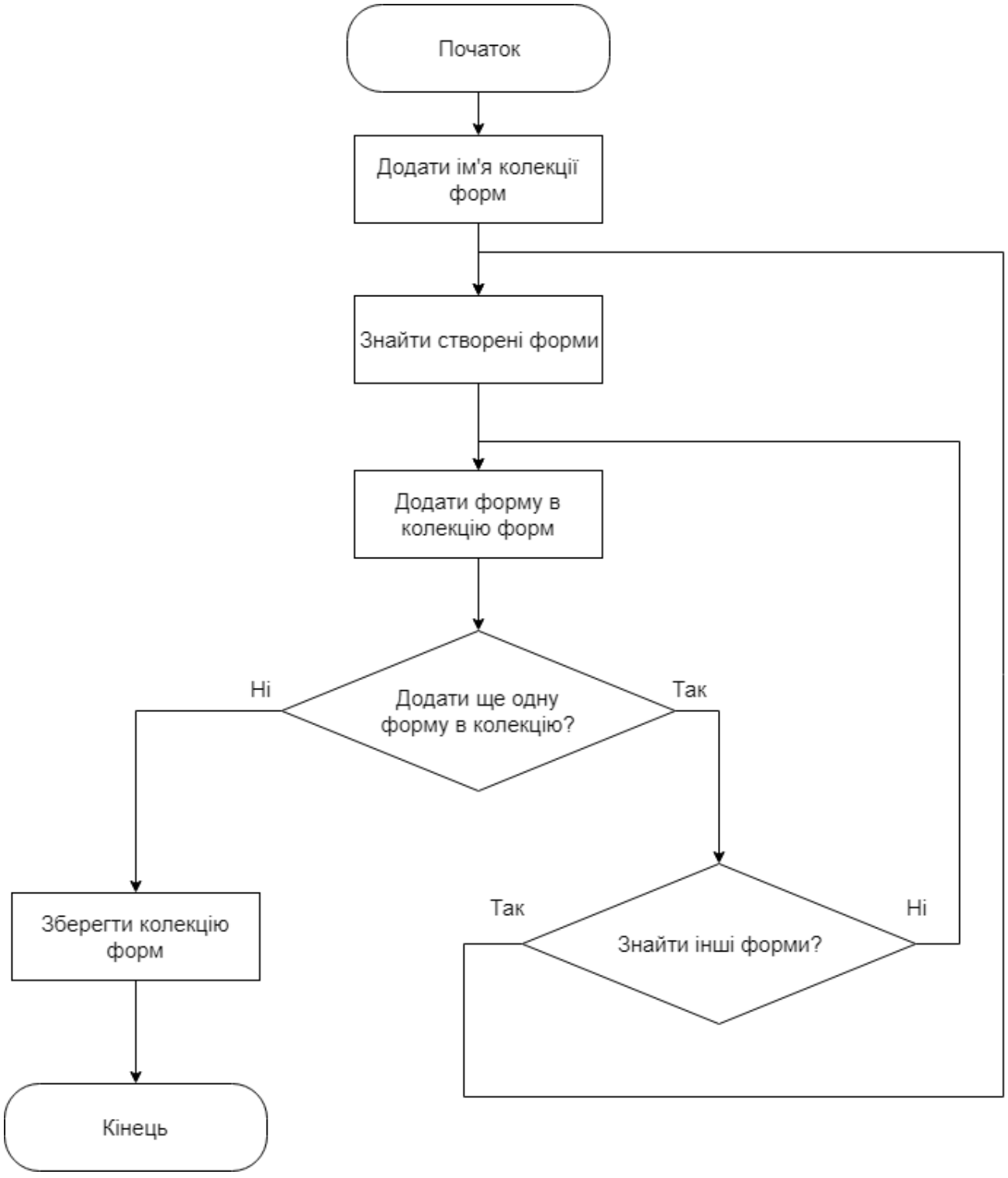

Рисунок 3.2 – Алгоритм створення колекції форм

При проходженні колекції форм, необхідно перейти за отриманим посиланням та відповісти на поставлені запитання. В ході проходження опитування, введені дані будуть валідуватися. Після проходження всіх форм перевірити введені дані та зберегти результат.

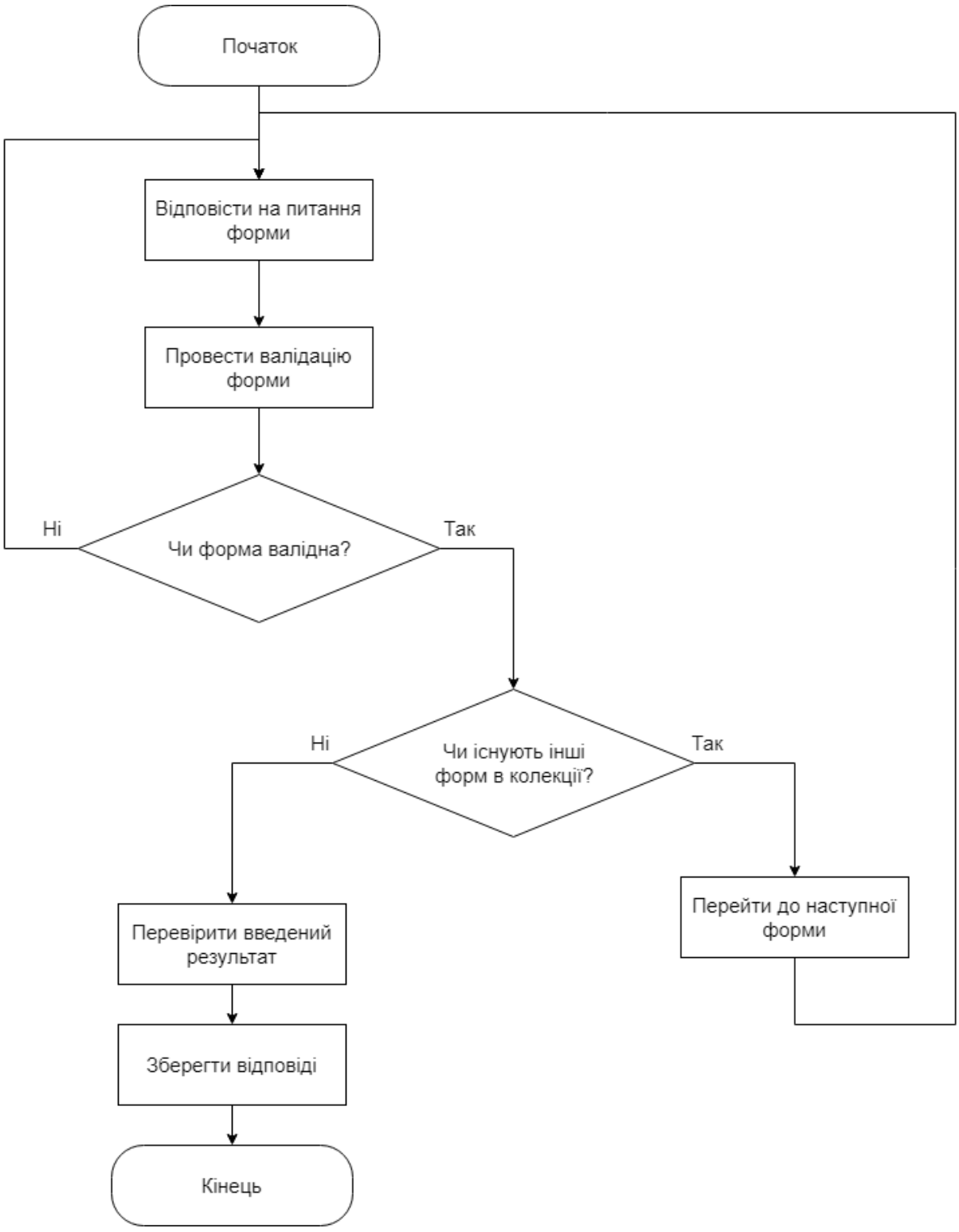

Рисунок 3.3 – Алгоритм відповіді на колекцію форм

Для аналізу отриманих даних від респондентів потрібно запустити алгоритм обробки даних.

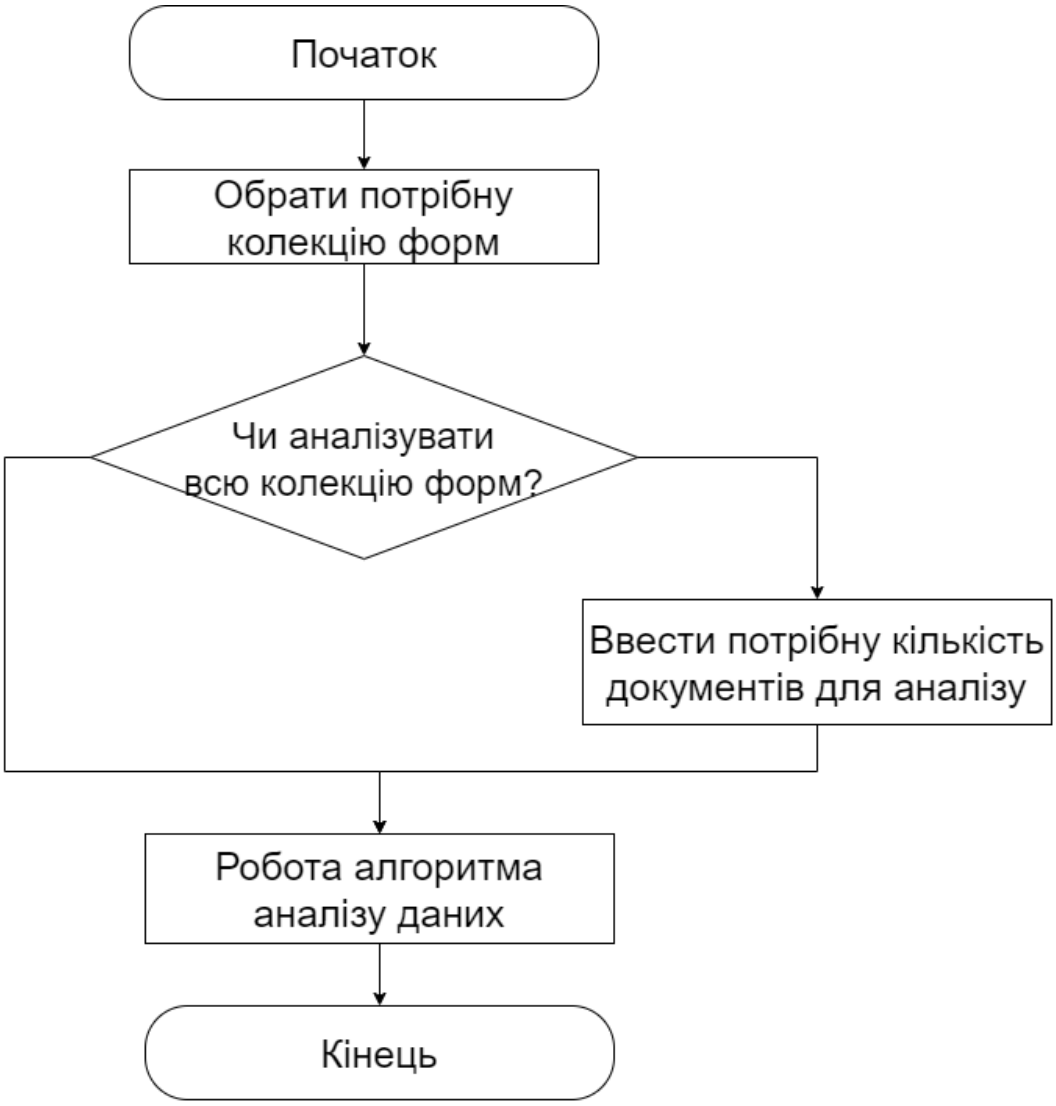

Рисунок 3.4 – Алгоритм запуску аналізу колекції форм

4 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

4.1 Діаграма використання

Для того, щоб більш детально ознайомитись з предметною областю, буде доречно використати діаграму прецедентів (Use-Case діаграму).

Use-Case діаграма – це динамічна діаграма або схема поведінки в UML. Використовується Use-Case діаграма для моделювання функціональності системи за допомогою акторів та випадків використання. Випадки використання – це сукупність дій, служб та функцій, які потрібно виконувати системі. У цьому контексті "система" – це щось, що розробляється чи експлуатується, наприклад, веб-сайт. "Актори" – це люди або організації, які працюють у визначених ролях у системі. [21].

Use-Case діаграми корисні для візуалізації функціональних вимог системи, що впливають на вибір дизайну та пріоритету розвитку. Вони також допомагають визначити будь-які внутрішні чи зовнішні фактори, які можуть впливати на систему, і їх слід враховувати.

Виходячи з отриманих ролей в попередньому розділі, та даних про їх права, можна виділити декілька Use-Case діаграм.

Рисунок 4.1 – Діаграма прецедентів для користувача «Гість»

На рисунку 4.1 зображена Use-Case діаграма для користувача «Гість». Гість може зареєструватися в системі, а якщо у нього вже є обліковий запис, то може ввійти в систему. Також може пройти колекцію форм, але тільки в тому випадку, якщо він має пряме посилання.

Рисунок 4.2 – Діаграма прецедентів для користувача «Користувач»

На рисунку 4.2 зображена Use-Case діаграма користувача «Користувач». Користувач може управляти своїми формами та колекціями форм (створювати, редагувати, видаляти). Користувач може прямо з системи відправити посилання на проходження колекції форм, а після отримання відповідей, експортувати їх до CSV файлу. Також користувач має можливість запустити алгоритм обробки даних.

Рисунок 4.3 – Діаграма прецедентів для користувача «Адміністратор»

На рисунку 4.3 зображена Use-Case діаграма для користувача «Адміністратор». Загалом, адміністратор має такі самі можливості, як і користувач, але може управляти не лише своїми формами та колекціями форм, але і інших користувачів. Такі права потрібні для того, щоб запобігати певних можливих махінацій в системі або поширенню забороненої інформації.

4.2 Бізнес ролі в системі

Загалом в системі потрібно виділити 3 логічні ролі користувачів – гість, користувач та адміністратор.

Таблиця 4.1 – Бізнес-ролі в системі

Бізнес-роль	Коротке ім’я	Функції
Незареєстрований користувач системи	Гість	Проходження колекції форм за отриманим посиланням, реєстрація, аутентифікація та авторизація.
Зареєстрований користувач системи	Користувач	Займається створенням, редагуванням, видаленням та перегляд форм та колекцій форм. Може запросити респондента пройти колекцію форм. Може запустити алгоритм обробки даних.
Зареєстрований адміністратор системи	Адміністратор	Має повний доступ до всіх форм та колекцій форм, а також функцій управління ними.

4.3 Вимоги до програмного додатка

Оскільки сервіс повинен працювати з персональними даними користувачів, то:

– Необхідно використовувати захищену, відмовостійку базу даних;

– Необхідно застосувати шифрування для особливо важливих даних, щоб запобігти їх захоплення зловмисниками;

– Необхідно проводити ретельний аналіз роботи адміністраторів;

– Необхідно підтримувати логування подій системи.

– Необхідно дотримуватися закону і підзаконні акти щодо персональних даних користувачів.

4.4 Проектування інтерфейсу користувача

Користувальницький інтерфейс є своєрідним комунікаційним каналом, по якому здійснюється взаємодія користувача і комп'ютера.

Кращий користувальницький інтерфейс – це такий інтерфейс, якому користувач не повинен приділяти багато уваги, майже не помічати його. Користувач просто працює, замість того, щоб розмірковувати, яку кнопку натиснути або де клацнути мишею. Такий інтерфейс називають прозорим – користувач ніби дивиться крізь нього на свою роботу.

Щоб створити ефективний інтерфейс, який робив би роботу з програмою приємною, треба розуміти, які завдання будуть вирішувати користувачі з допомогою даної програми і які вимоги до інтерфейсу можуть виникнути у користувачів.

При проектуванні інтерфейсу користувача, потрібно дотримуватися три основні положення:

1. програма повинна допомагати виконати завдання, а не ставати цим завданням;
2. при роботі з програмою користувач не повинен відчувати себе дурнем;
3. програма повинна працювати так, щоб користувач не вважав комп'ютер дурнем.

На рисунку 4.4 зображено шаблон сторінки створення текстового поля. Ця сторінка буде мати такі поля:

– назва поля;

– значення за замовчуванням;

– помітка, чи поле обов’язкове;

– помітка, чи поле зашифроване;

– та поле для регулярного вираження.

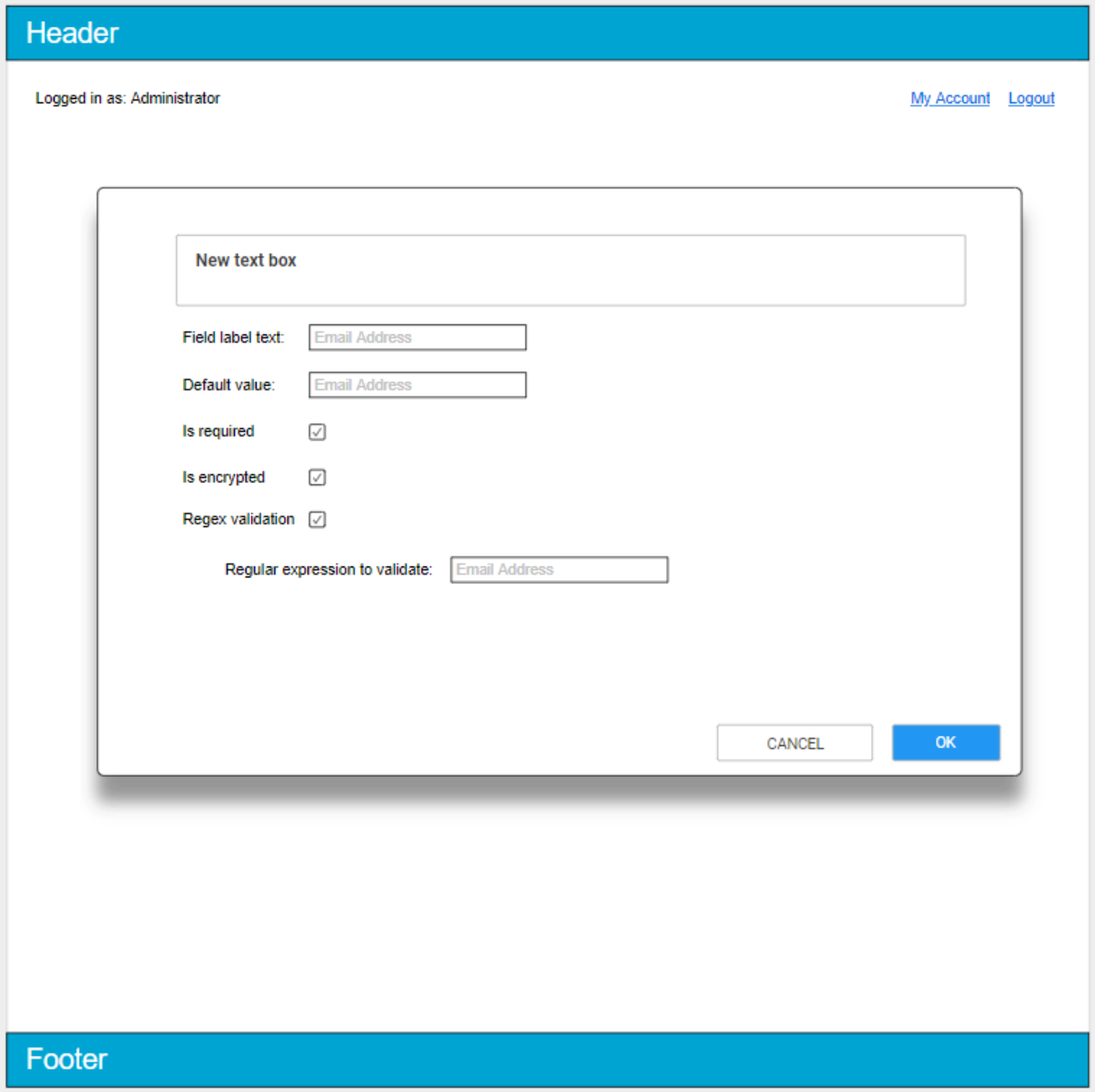

Рисунок 4.4 – Макет сторінки створення та редагування textbox

На рисунку 4.5 зображено шаблон створення випадаючого списка. Ця сторінка буде мати такі поля:

– назва поля;
– помітка, чи поле обов’язкове;
– текс та значення елемента списка;
– поле для сортування та видалення елементів списку.

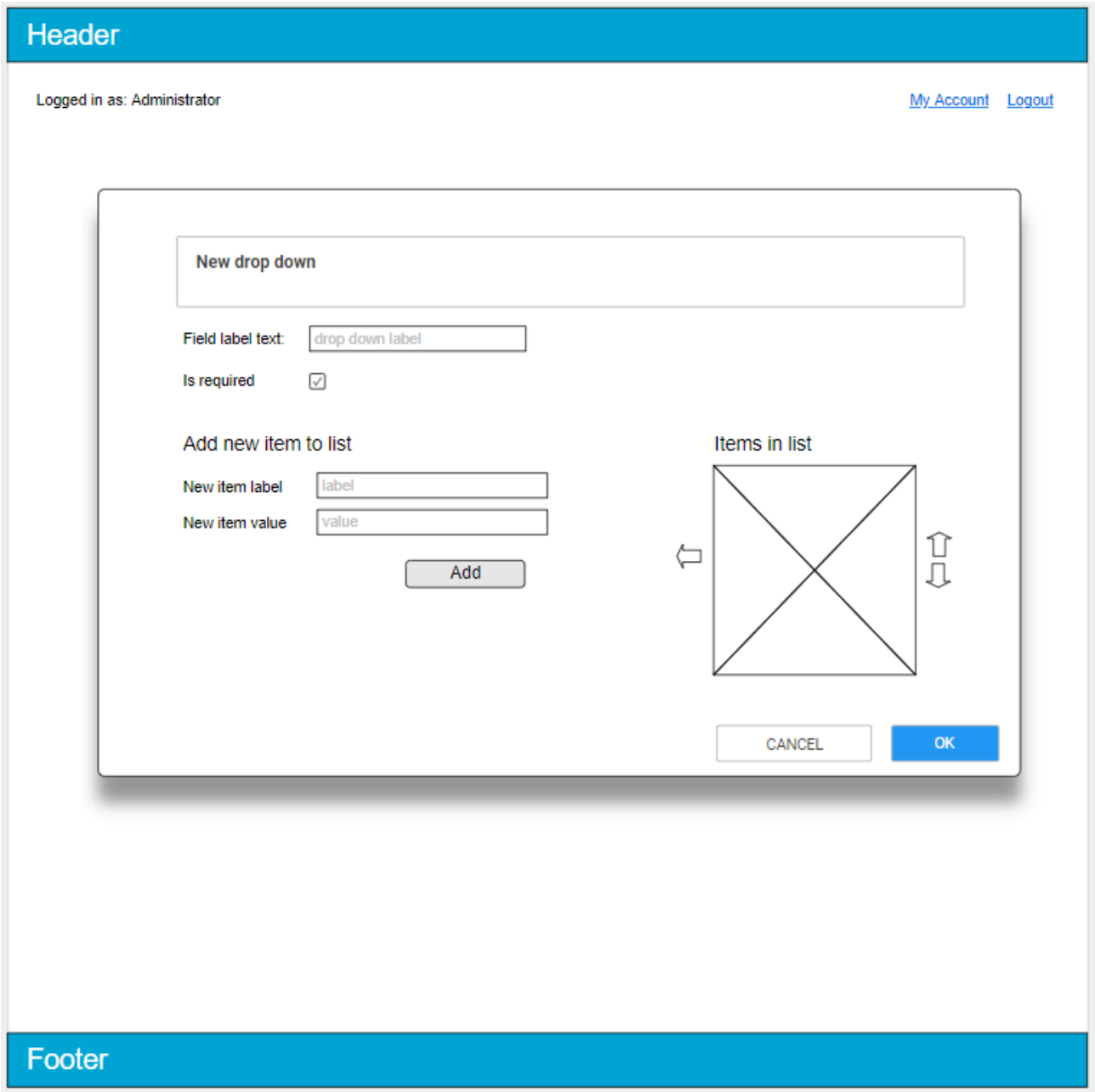

Рисунок 4.5 – Макет сторінки створення та редагування dropdown

На рисунку 4.6 зображено шаблон сторінки створення текстової області. Ця сторінка буде мати такі поля:

– назва поля;
– значення за замовчуванням;
– кількість рядків;
– помітка, чи поле обов’язкове.

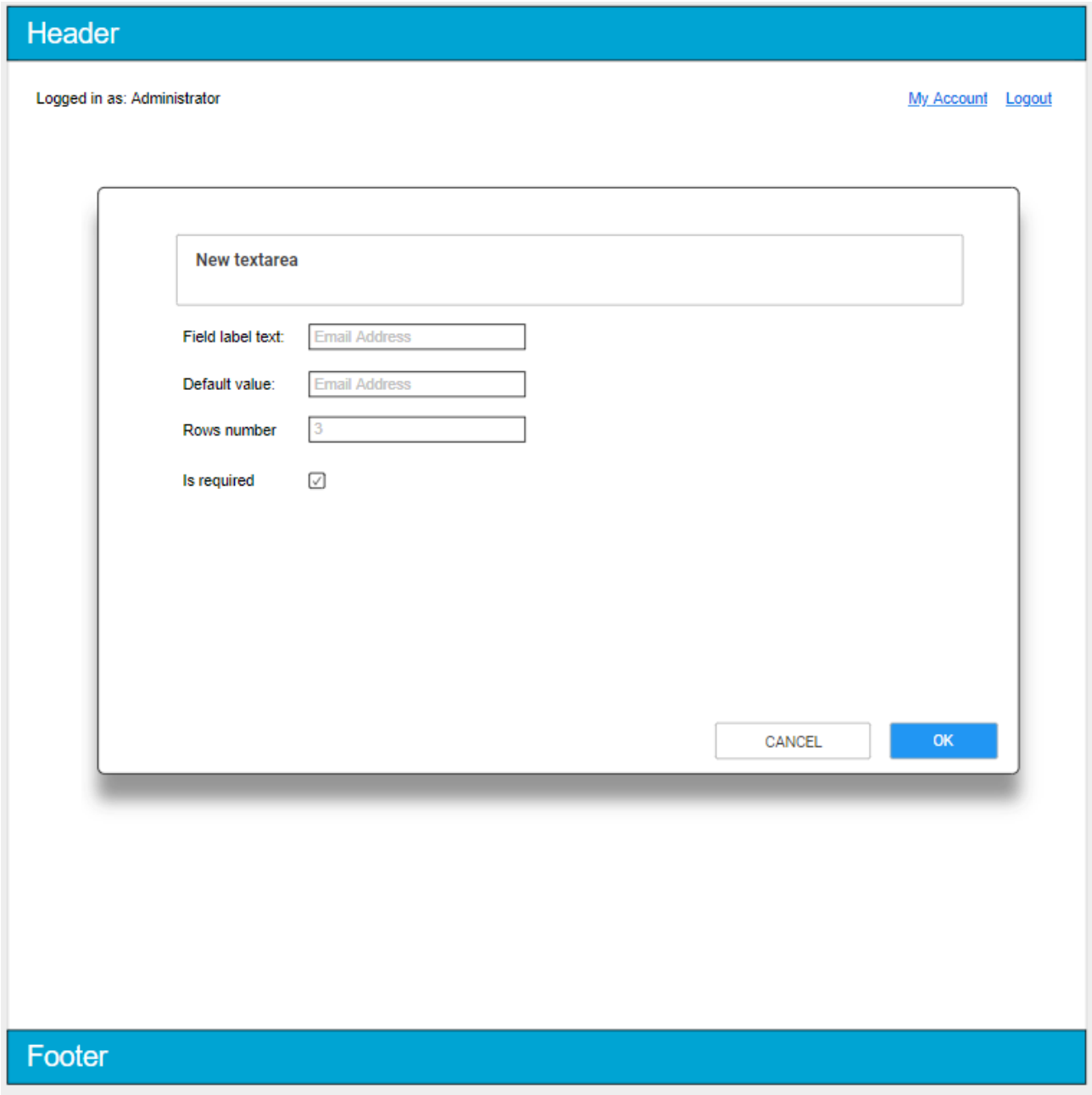

Рисунок 4.6 – Макет сторінки створення та редагування textarea

На рисунку 4.7 зображено шаблон сторінки створення checkbox list та radio group (перемикачів). Ця сторінка буде мати такі поля:

– назва поля;
– текс та значення елемента списка;
– поле для сортування та видалення елементів списку.

При введенні даних в поля тексту та значення елемента управління, опція потрапляє в праву частину екрана і буде відображатися так, як це буде при проходженні форми.

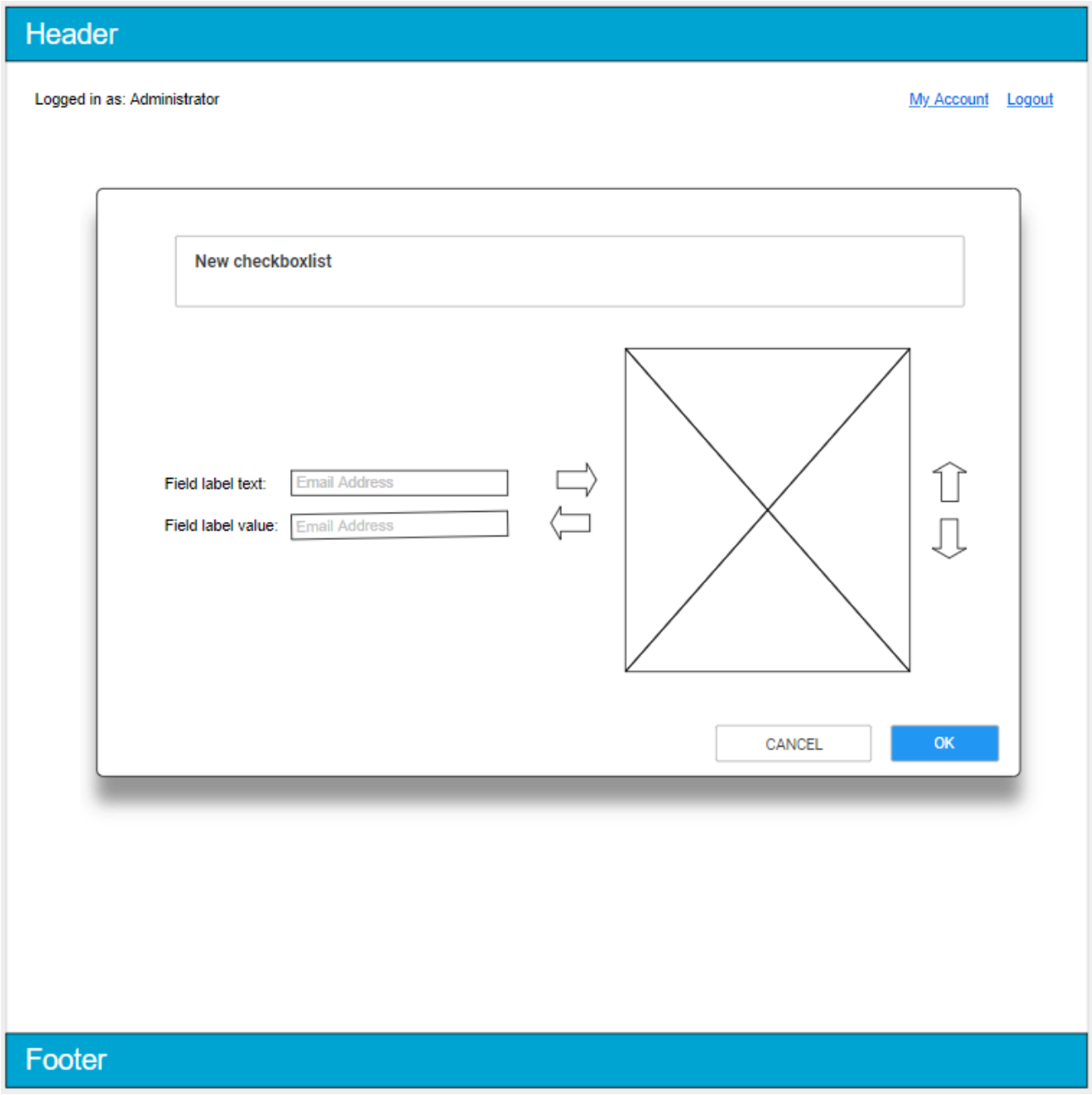

Рисунок 4.7 – Макет сторінки створення та редагування checkbox list та radio group

На рисунку 4.8 зображено шаблон сторінки створення елемента управління датою. Ця сторінка буде мати такі поля:

– назва поля;
– значення за замовчуванням;

– помітка, чи поле обов’язкове;
– мінімальна та максимальна дати.

При створенні елементу datepicker ми повинні ввести мінімальну та максимальну дати. Це буде обмежувати введене значення в елемент управління. Також потрібно передбачити введення тільки максимальної або мінімальної дати.

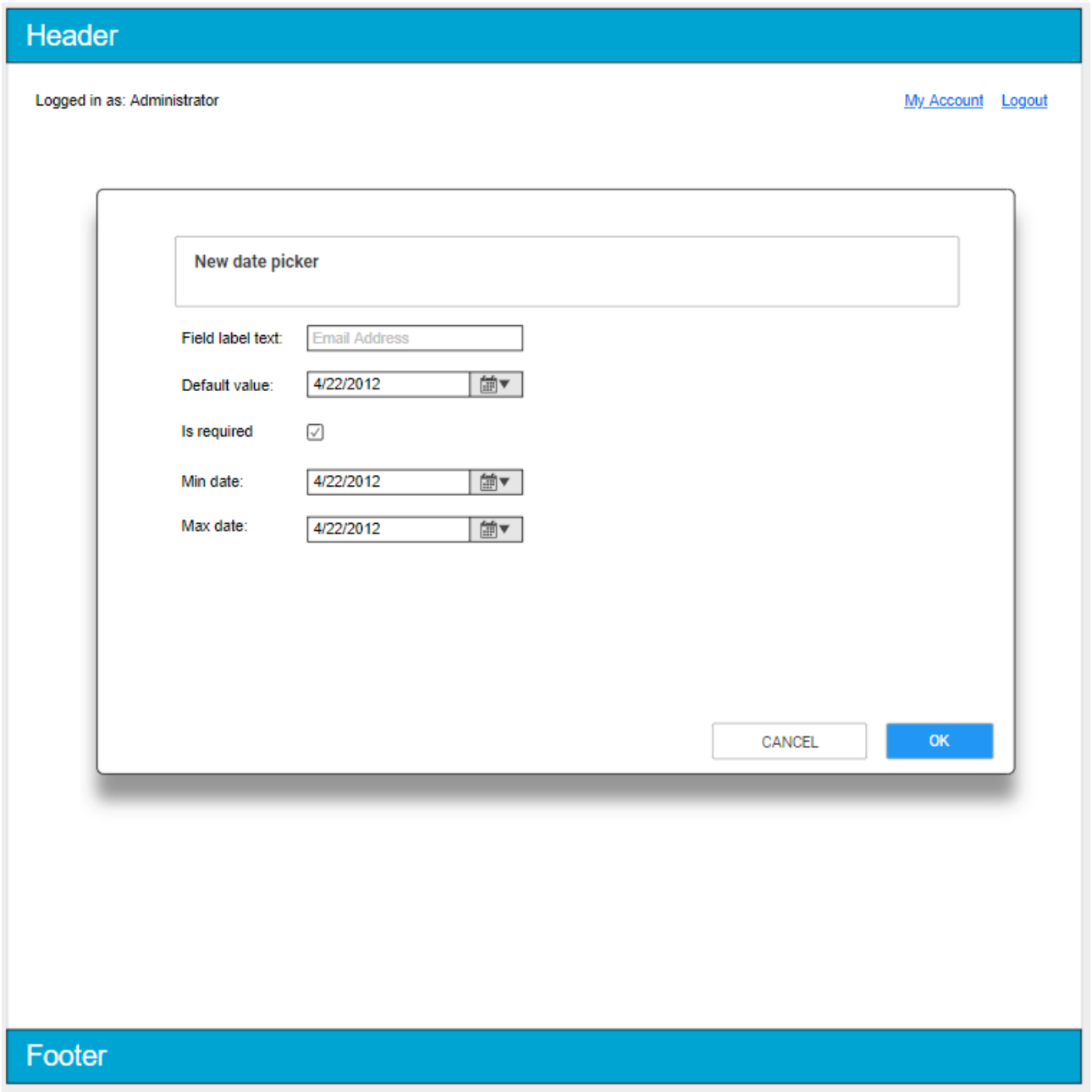

Рисунок 4.8 – Макет сторінки створення та редагування datepicker

На рисунку 4.9 зображено шаблон сторінки перегляду та проходження створеної колекції форм. Ця сторінка буде поділена на дві частини: список форм в колекції форм та перегляд поточної форми.

При натисканні на елемент списку форм в колекції форм, в правій частині буде змінюватися форма. При введені невірних даних потрібна відбуватися валідація форми.

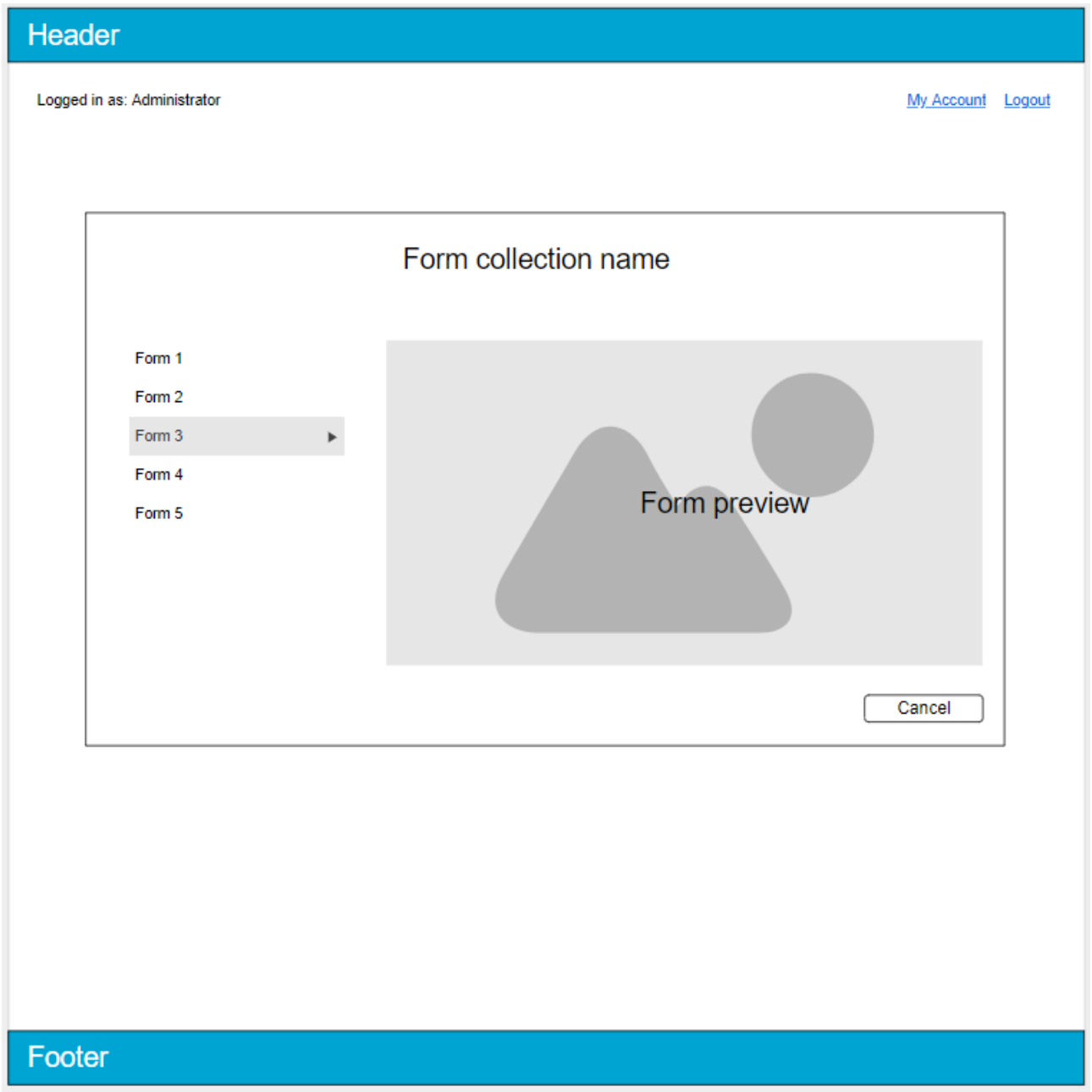

Рисунок 4.9 – Макет сторінки перегляду та проходження колекції форм

На рисунку 4.10 зображено шаблон сторінки перегляду введених відповідей на колекцію форм. Ця сторінка буде мати окремі логічні блоки (поділені за формами в колекції форм) з назвою поля та відповіддю на конкретне запитання.

Також буде можливість змінити введені дані або роздрукувати результат введення.

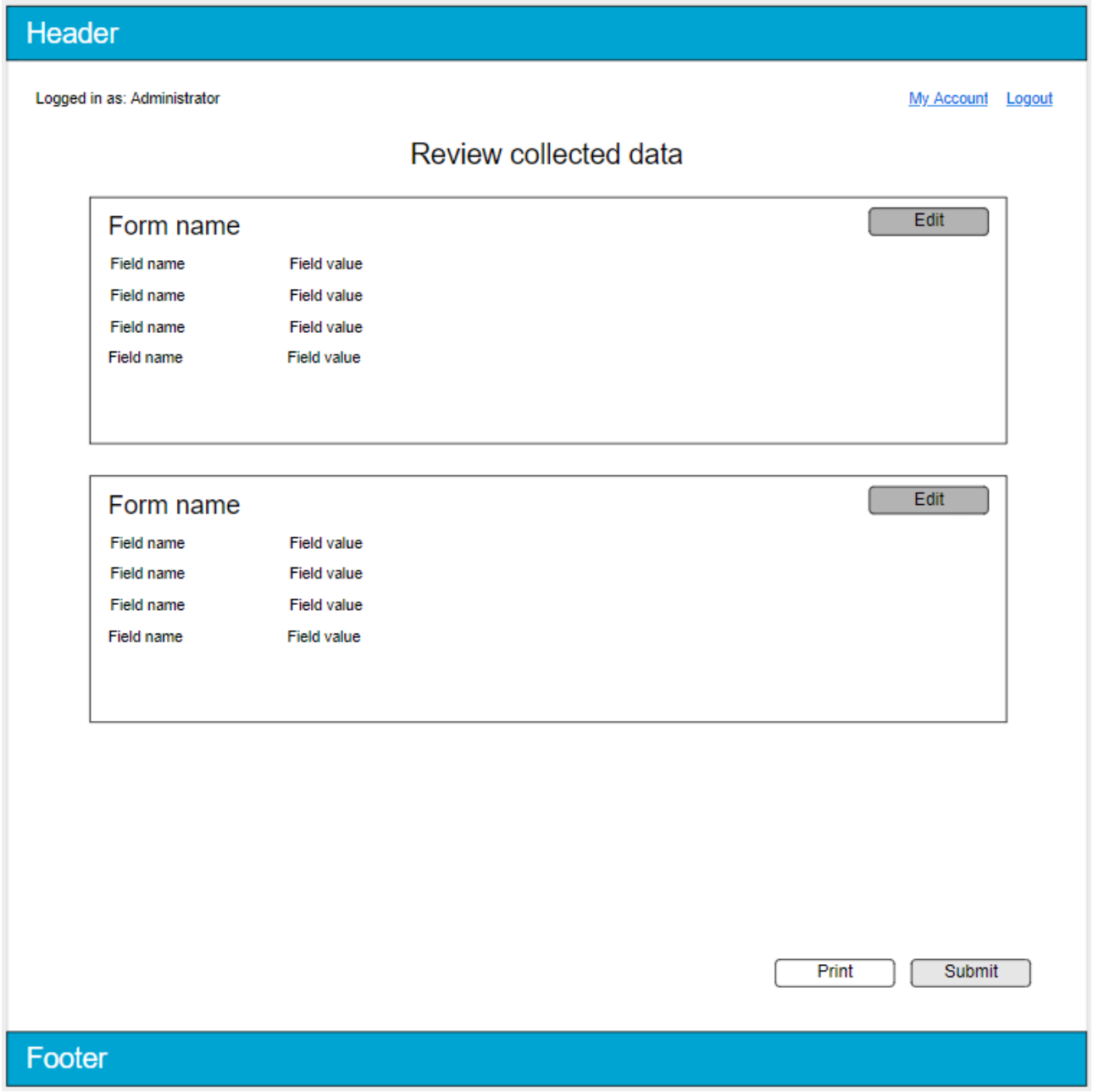

Рисунок 4.10 – Макет сторінки перегляду результатів

5 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ

5.1 Обґрунтування вибору мов програмування

5.1.1 Back-End

Для розробки серверної частини сервісу, було вирішено використовувати фреймворк ASP.NET Core з використанням мови C# [24][25][26][27].

ASP.NET Core – cross-платформне, високопродуктивне середовище з відкритим вихідним кодом для створення сучасних cloud та веб-додатків[27].

Переваги:

- єдине рішення для створення користувацького веб-інтерфейсу і веб-API;
- інтеграція сучасних клієнтських платформ і робочих процесів розробки;
- cloud-система конфігурації на основі середовища;
- вбудоване введення залежностей;
- спрощений високопродуктивний модульний конвеєр HTTP-запитів;
- можливість розміщення в IIS, Apache, Docker або у власному процесі;
- інструментарій, що спрощує процес сучасної веб-розробки;
- можливість побудови та запуску в ОС Windows, macOS і Linux;
- відкритий вихідний код і орієнтація на співтовариство.

Даний фреймворк дозволить створити потужний веб-додаток, з можливістю розповсюдження на будь-яку платформу[28][29].

Для обробки даних було використано технологію Azure Function App. Azure Function - це обчислювальна служба без сервера, яка дозволяє користувачеві запускати код, що викликається подіями, не потребуючи надання та управління інфраструктурою. Будучи службою на основі тригера, він виконує сценарій або фрагмент коду у відповідь на різноманітні події.

Функції Azure можна використовувати для досягнення decoupling, високої пропускної здатності, повторного використання та спільного використання. Будучи більш надійним, його можна використовувати і для виробничих середовищ.

Функції Azure підтримують різні мови, такі як C #, F #, JavaScript, node.js та багато іншого.

5.1.2 Front-End

При виборі інструменту розробки інтерфейсу користувача, було розглянуто дві основні альтернативи: React та Angular.

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript для побудови користувальницьких інтерфейсів. Він підтримується Facebook та спільнотою окремих розробників та компаній.

React можна використовувати як базу при розробці односторінкових або мобільних додатків. Однак React стосується лише надання даних у DOM, тому створення React додатків зазвичай вимагає використання додаткових бібліотек для управління станом та маршрутизації. Redux та React Router є відповідними прикладами таких бібліотек.

React значно полегшує створення інтерфейсів завдяки розділенню кожної сторінки на невеликі фрагменти – компоненти.

Головні можливості ReactJS:

– використання JSX для створення шаблону, замість звичайного JavaScript;
– React Native;
– Virtual Document Object-Model.

Переваги:

– простий для вивчення;
– можливість повторного використання компонентів;
– потужні засобі для розробки.

Недоліки:

– погана документація;
– мале співтовариство.

Angular - це платформа та фреймворк для побудови односторінкових клієнтських додатків за допомогою HTML та TypeScript. Angular написаний на мові програмування TypeScript. Він реалізує основну та додаткову функціональність як набір бібліотек TypeScript, які ви імпортуєте у свої програми.[34]

Головні можливості:

– вбудований DI;
– компонентна модель;
– Angular CLI.

Переваги:

- two-way data binding;
- директиви;
- dependency injection;
- використання мови TypeScript;
- велике співтовариство.

Недоліки:

- продуктивність;
- складний для вивчення.

Оскільки Angular являє собою цілий фреймворк, було вирішено використовувати само його. Це дозволить розробити продукт з досить потужним інтерфейсом користувача.

5.2 Обґрунтування вибору архітектури

5.2.1 Архітектура Back-End

Існує безліч різних видів і типів архітектури, які успішно застосовуються. Однієї з найбільш використовуваних є класична трирівнева система, яка передбачає поділ додатка на три рівні: доступ до бази даних, логіка додатка та інтерфейс користувача [18]. В рівні інтерфейсу користувача буде використано шаблон проектування MVC – Model-View-Controller [27].

Були виділені такі рівні:

- Data Access Layer
- Business Layer
- Presentation Later

Presentation layer (рівень представлення) – рівень, з яким безпосередньо взаємодіє користувач. Цей рівень включає компоненти для інтерфейсу користувача, механізм отримання даних від користувача. Стосовно до ASP.NET на даному рівні розташовані view і всі ті компоненти, що складають для інтерфейс користувача (стилі, статичні сторінки html, javascript), а також моделі уявлень та контролери.

Business layer (рівень бізнес-логіки) – містить набір сервісів (класів), які реалізують всю необхідну бізнес-логіку для вирішення поставлених задач. Даний рівень бере на себе обрабку даних, що були отримані від Presentation layer.

Data Access layer (рівень доступу до даних) – сховище data-model об’єктів. Ці моделі описують сутності, що використовуються системою. Також в даному рівні зберігаються класи, що відповідають за взаємодію з базою даних. В нашому аипадку це рерозиторії і Unit of work.

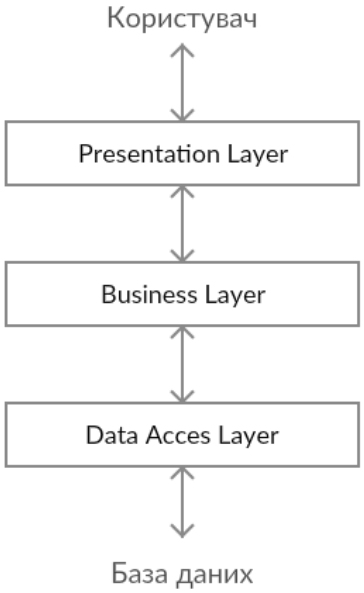

Рисунок 5.1 – Схема архітектури додатка

При проектуванні рівня представлення, було вирішено використовувати шаблон MVC. MVC – це модель дизайну додатків, що складається з трьох взаємопов’язаних частин. Вони включають модель (дані), вид (інтерфейс користувача) та контролер (процеси, що обробляють введення).

Модель або модель "MVC" зазвичай використовується для розробки сучасних інтерфейсів користувача. Він пропонує основні елементи для розробки програм для настільних або мобільних пристроїв, а також веб-додатків. Він добре працює з об'єктно-орієнтованим програмуванням, оскільки різні моделі, види та контролери можуть розглядатися як об'єкти та повторно використовуватися в програмі.

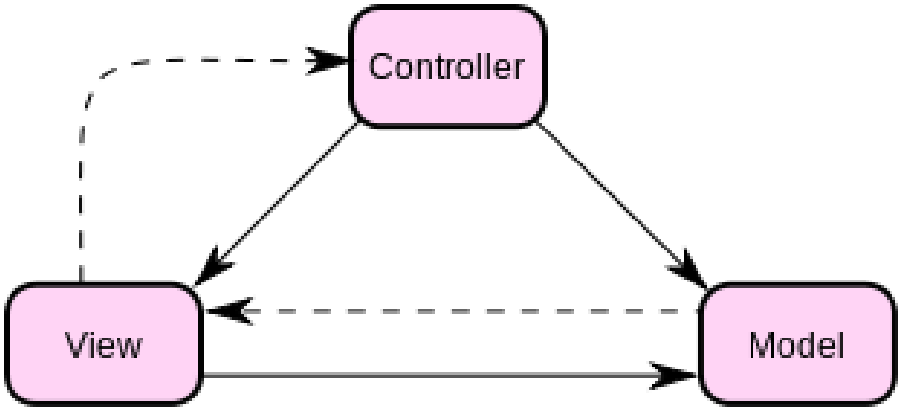

Рисунок 5.2 – Шаблон MVC

Модель - це дані, які використовує програма. Це може бути база даних, файл або простий об’єкт, наприклад, значок або символ у відеоіграх.

Представлення – це засіб відображення об'єктів у програмі. Приклади включають відображення вікна, кнопок або тексту у вікні. Він включає все, що може бачити користувач.

Контролер оновлює і моделі, і види. Він приймає введення та виконує відповідне оновлення. Наприклад, контролер може оновлювати модель, змінюючи атрибути персонажа у відеоігри. Він може змінити подання, відобразивши оновлений символ у грі.

Мета шаблону – допомогти нам відокремити різні аспекти програми (бізнес-логіка, графічний інтерфейс), забезпечуючи при цьому зв'язок між цими елементами. Таким чином, інформаційна (найбільш багаторазова) логіка належить до моделі, GUI належить до представлення. Логіка вводу належить до контролера. Цей поділ допомагає керувати складністю під час створення програми, оскільки вона дозволяє зосередити увагу на одному аспекті реалізації одночасно.

5.2.2 Function App

Функції Azure дозволяють запускати невеликі фрагменти коду (звані "функції"), не турбуючись про інфраструктуру додатків. За допомогою функцій Azure хмарна інфраструктура надає всі сучасні сервери, необхідні для того, щоб програма працювала в масштабі.

Функцію "розвочинає (trigger)" конкретний тип події. Підтримувані тригери включають реагування на зміни в даних. Azure Functions має інтуїтивно зрозумілий

інтерфейс на основі браузера для реагування на події, породжені Event Hubs, HTTP запитом, Timers, Azure Queues, Table Storage, Blob Storage тощо.

Функції – чудове рішення для обробки об'ємних даних, інтеграції систем, роботи з Інтернетом речей (IoT) та побудови простих API та мікросервісів.

5.2.3 Взаємодія Back-End та Front-End

Спілкування серверної та клієнтської частин систему відбувається за допомогою HTTP запитів. Клієнтський додаток відправляє запит на API сервера та в свою чергу отримує від нього відповідь з необхідними даними [30][31][32]. API в свою чергу буде додатково спілкуватися з Azure Function. Azure Function беспосередньо відповідає за обробку даних. Для цього він має доступ до бази даних для зчитування отриманих даних та для запису результатів обробки.

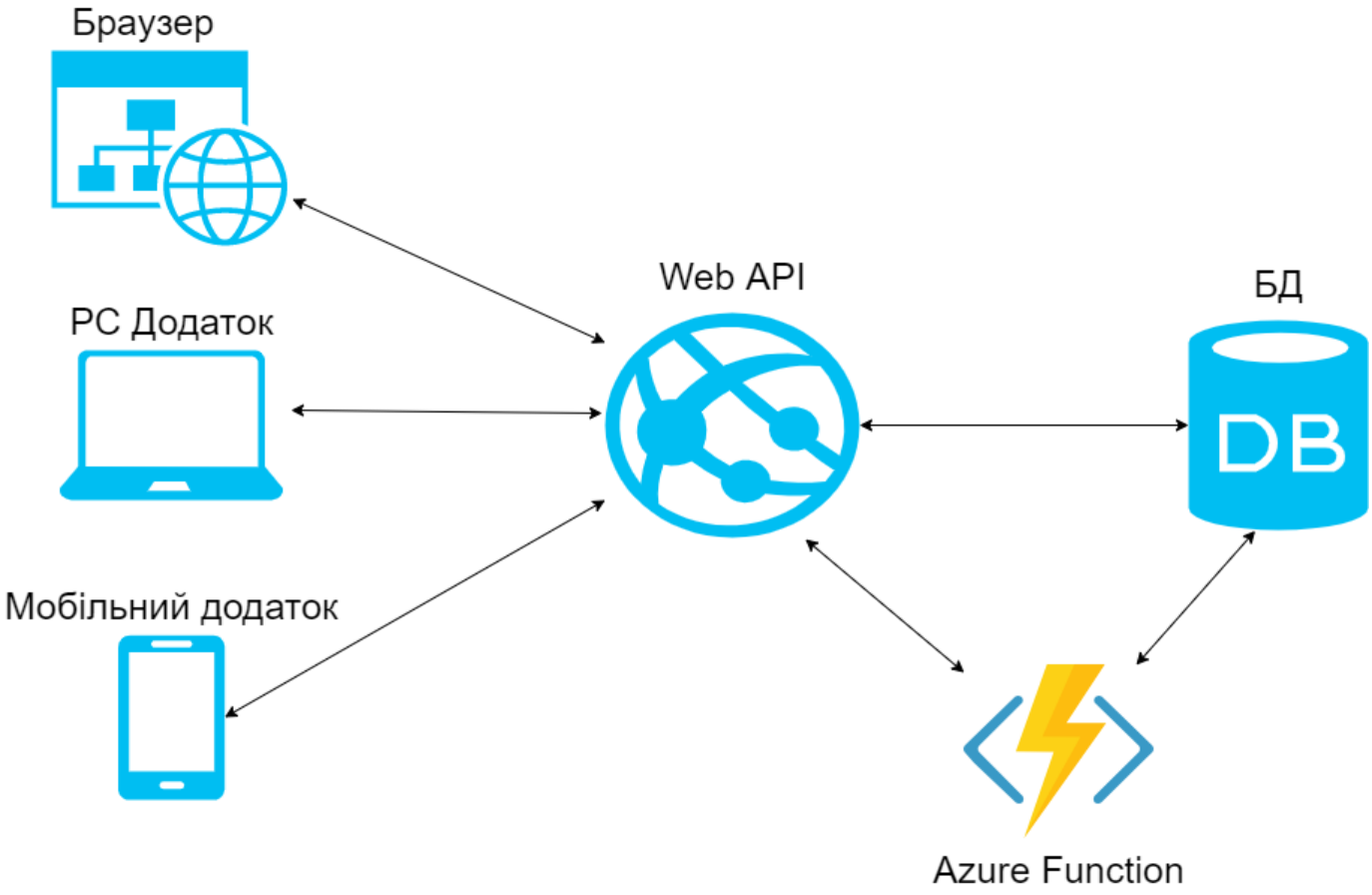

Рисунок 5.3 – Узагальнена схема роботи Web API

5.3 Обґрунтування вибору СУБД

MongoDB – це система управління базами даних (СУБД) з відкритим кодом, яка використовує документно-орієнтовану модель даних, яка підтримує різні форми даних. Це одна з численних нереляційних технологій баз даних, що виникла в середині 2000-х під банером NoSQL для використання у Big Data програмах та інших роботах з обробки даних, що включають дані, що не вписуються в жорстку реляційну модель. Замість використання таблиць та рядків, як у реляційних базах даних, архітектура MongoDB складається з колекцій та документів.

Основні можливості MongoDB:

– документо-орієнтована;
– мова запитів схожа на JavaScript;
– запити можуть бути динамічні;
– підтримка індексів;
– зберігання бінарних даних;
– журналювання;
– асинхронна реплікація;
– набір реплік;
– шардінг.
– може працювати відповідно до парадигми MapReduce.

Переваги:

– швидкість;
– sharding;
– гнучкість;
– висока доступність;
– масштабованість;
– легкість у встановленні;
– спеціальна підтримка запитів.

Недоліки:

- високе споживання пам’яті;
- обмеження на розмір документу;
- неможливість об’єднання даних різних документів при запиті.

При виборі БД, необхідно в першу чергу брати до уваги можливе навантаження та структуру даних, які необхідно зберігати. MongoDB зберігає дані у вигляді BSON-документів, що дозволяє без зусиль зберігати неоднорідні дані. Також MongoDB надає великі можливості по розподіленню навантаження. Тому було вирішено обрати MongoDB як засіб зберігання даних.

5.4 Моделювання даних системи

Для зберігання даних в MongoDB використовуєтьсятип, що називається BSON (binary JSON).

На відміну від реляційних баз даних в MongoBD можна зберігати різноманітні об’єкти (не залежно від властивостей та структури).

Так як MongoDB являється документо-орієнтованою базою даних, то вона зберігає документи, на відміну від рядків в таблицях. Документи зберігаються в колекціях. В свою чергу документ – це сховище ключів і значень.

Рис 5.4 – Колекція в MongoDB

Наприклад, документ форми буде мати такий вигляд:


```
{
    "_id": "KU+rcKy4QiS4CEoYj14oyQ==",
    "_t": ["FormBase", "DynamicForm"],
    "IsDeleted": false,
    "name": "Starlor",
    "lastModified": "2020-05-12T08:53:54.799Z",
    "isValid": false,
    "type": 1,
    "isMultiEntry": false,
```

```
        "controlSets": [],
        "controls": [{
                "_t": ["Control", "TextBox"],
                "_id": "Uy7bpRyrztwR+s2ANF2g0A==",
                "IsDeleted": false,
                "type": "textbox",
                "label": "Text field",
                "validationRules": [{
                        "type": 0,
                        "value": null,
                        "message": "This field is required."
                }, {
                        "type": 1,
                        "value": null,
                        "message": null
                }],
                "value": "qwe"
        }, {
                "_t": ["Control", "CheckBoxGroup"],
                "_id": "9lY8lXjP3m21YfDk7OZWyQ==",
                "IsDeleted": false,
                "type": "checkbox",
                "label": "Checkbox list",
                "validationRules": [],
                "options": [{
                        "name": "qwe",
                        "value": "qwe",
                        "isChecked": false
                }],
                "values": []
        }]
}
```


А документ колекції форм:

{
"_id": "y0gGVvDSTqOLOuMiKPs0LA==",
"_t": ["BaseEntity", "FormCollection"],
"IsDeleted": false,
"name": "fc1",
"lastModified": "2020-05-12T08:54:42.849Z",
"submissionUrl": "",
"forms": ["/tF1L7jrToSU6bU8dyly2A==", "KU+rcKy4QiS4CEoYj14oyQ=="]
}

5.5 Шаблони доступу до даних

Для доступу до даних були використані шаблони Repository та Unit of Work.

Репозиторій представляє патерн, завдання якого полягає в управлінні доступом до джерела даних. Клас, який реалізує даний патерн, не містить бізнес-логіку, не керує бізнес-процесами, він тільки містить операції над даними. Як правило, репозиторій реалізує CRUD-інтерфейс, тобто представляє операції по вилученню, додаванню, редагування і видалення даних.

Прикладом може бути інтерфейс репозиторія:

```
public interface IGenericRepository<TEntity> where TEntity : class
   {
      void Create(TEntity item);
      TEntity FindById(int id);
      IEnumerable<TEntity> Get();
      IEnumerable<TEntity> Get(Func<TEntity, bool> predicate);
      void Remove(TEntity item);
      void Update(TEntity item);
   }
```

Якщо репозиторії використовують одне підключення до бази даних, то нерідко для організації доступу до одного підключення для всіх репозиторіїв додатки використовуютья інший патерн - Unit Of Work. Клас, який реалізує даний

патерн, як правило, містить набір репозиторіїв і ряд деяких загальних для них функцій.

5.6 Захист інформації в системі

Для реалізації аутентифікації та авторизації, була використана технологія ASP.NET Identity. ASP.NET Core Identity – це membership система, яка дозволяє додати функцію входу у вашу програму. Користувачі можуть створити обліковий запис та увійти з іменем користувача та паролем, або вони можуть за допомогою SSO скористатися зовнішніми постачальниками даних, такими як, Google, акаунт Microsoft, Twitter, Facebook та багато іншого.

Для аутентифікації запитів користувача використовується JWT (або JSON Web Token). JWT – це відкритий стандарт (RFC 7519), який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. Ця інформація може бути перевірина і їй можна довіряти, оскільки вона підписана цифровим шляхом.

Складається з 3-х частин: заголовку, даних та підпису.

Рисунок 5.5 – Розшифрований JWT

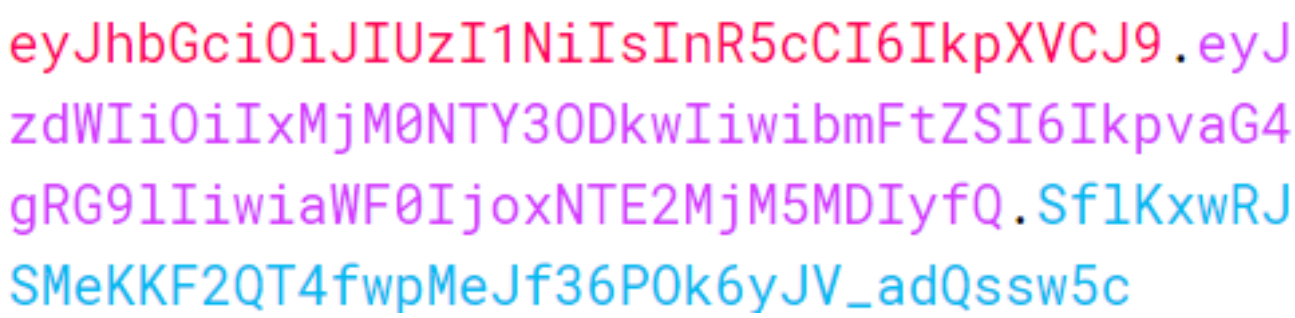

Рисунок 5.6 – Зашифрований JWT

5.7 Розгортання системи

Розгортання програмного засобу відбулося в системі Microsoft Azure. Microsoft Azure – це послуга хмарних обчислень, створена корпорацією Майкрософт для побудови, тестування, розгортання та управління програмами та службами через центри обробки даних, керовані Microsoft. Він надає програмне забезпечення як послугу (SaaS), платформу як послугу (PaaS) та інфраструктуру як послугу (IaaS) та підтримує безліч різних мов програмування, інструментів та фреймворків, включаючи як Microsoft, так і програмне забезпечення та системи сторонніх виробників. Як хмарне рішення, Azure є внутрішньо гнучким - він може створювати резервні копії даних практично будь-якою мовою, на будь-якій ОС та з будь-якого місця.

Завдяки автоматичному керуванню патчами для ваших віртуальних машин ми витрачаємо менше часу на управління інфраструктурою та можемо зосередитись на вдосконаленні додатка. Azure також пропонує постійну підтримку розгортання, що дозволяє впорядкувати поточні оновлення коду.

AutoScale – це функція, вбудована в веб-додатки Azure, яка автоматично налаштовує ресурси на основі веб-трафіку клієнта, щоб мати необхідні ресурси, коли трафік великий, і навпаки.

API проект був розгорнутий в Azure App Service. Azure App Service – це сервіс на основі HTTP для розміщення веб-додатків, API REST та back-end для мобільних пристроїв. Ми можеморозгортати додатки, написані будь-якою мовою програмування, будь то .NET, .NET Core, Java, Ruby, Node.js, PHP або Python. Програми запускаються та масштабуються з легкістю як у Windows, так і на базі Linux.

Azure App Service надає такі можливості, як безпека, балансування завантаження, автоматичне масштабування та автоматизоване управління. Ми також можемо скористатися його можливостями DevOps, такими як безперервне розгортання з Azure DevOps, GitHub, Docker Hub та іншими джерелами, управління пакетами, додаткові середовища, користувацький домен та сертифікати TLS / SSL.

Проект функцій був розгонутий в Azure Functions.

Azure Cosmos DB використовували як сховище данних. Даний сервіс - це глобально розповсюджена в усьому світі модель баз даних Microsoft. Cosmos DB дозволяє еластично та незалежно масштабувати пропускну здатність та налаштовувати зберігання в будь-якій кількості регіонів Azure по всьому світу. Головною перевагою є те, що ми можемо використовувати різні API, включаючи: SQL, MongoDB, Cassandra, Tables чи Gremlin. Cosmos DB надає комплексні гарантії про рівень обслуговування (SLA) для пропускної спроможності, затримки, доступності та узгодженості, чого не надають інші сервіси баз даних.

Також є режим «автопілота» , який дозволяє автоматично збільшувати або зменшувати заявлену пропускну здатність в залежності від навантаження на базу даних.

Політика індексування в Azure Cosmos DB може бути змінена в будь-який момент

5.8 Екранні форми додатка

Після проектування програмного засобу, почався етап активної розробки. Було розроблено базу даних с необхідною структурою, розроблена серверна та клієнтська частини додатку. Було розроблено всі поставленні задачі та отримали повністю функціонуючий, готовий до використання програмний продукт.

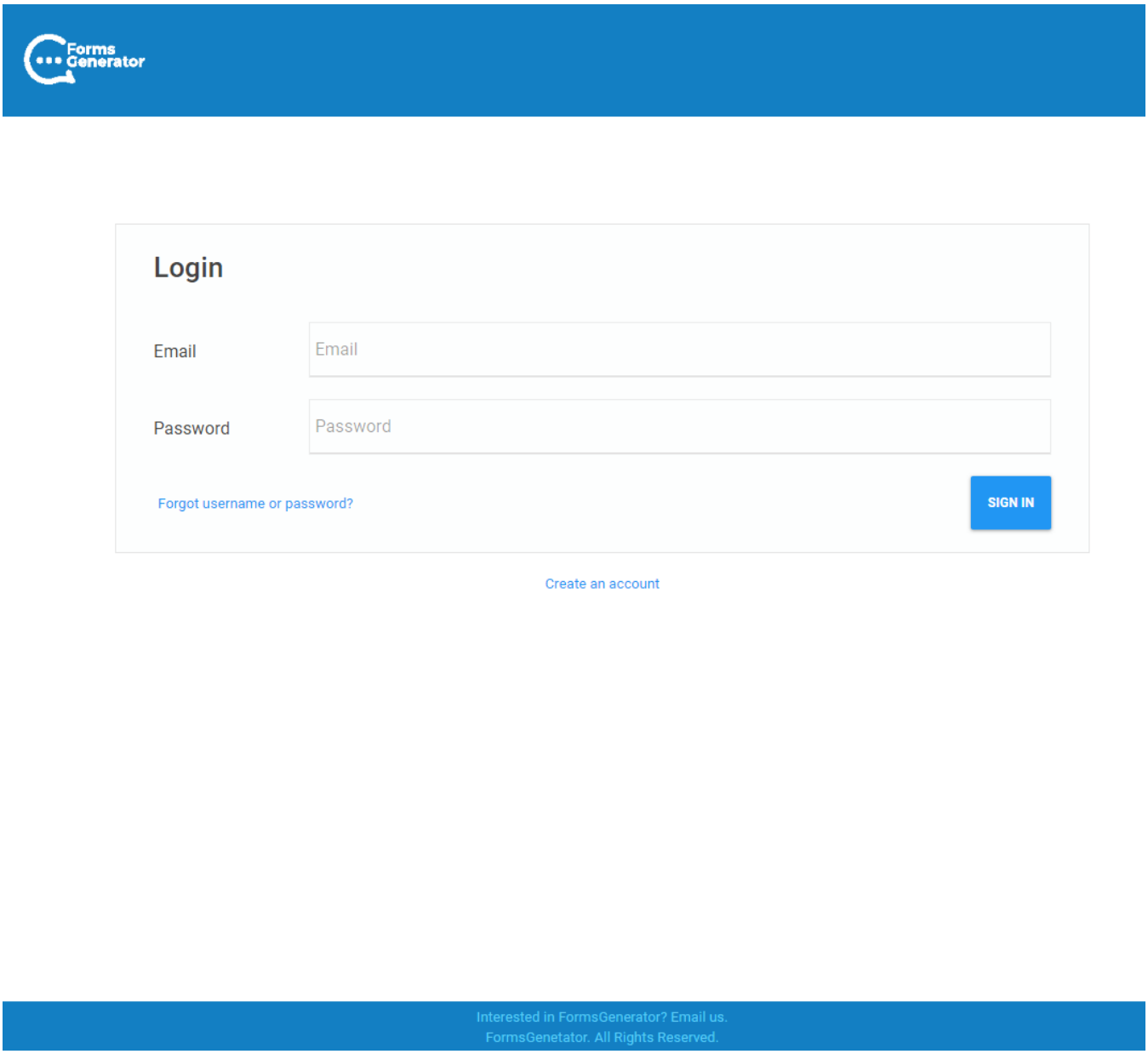

Рисунок 5.7 – Сторінка входу в систему

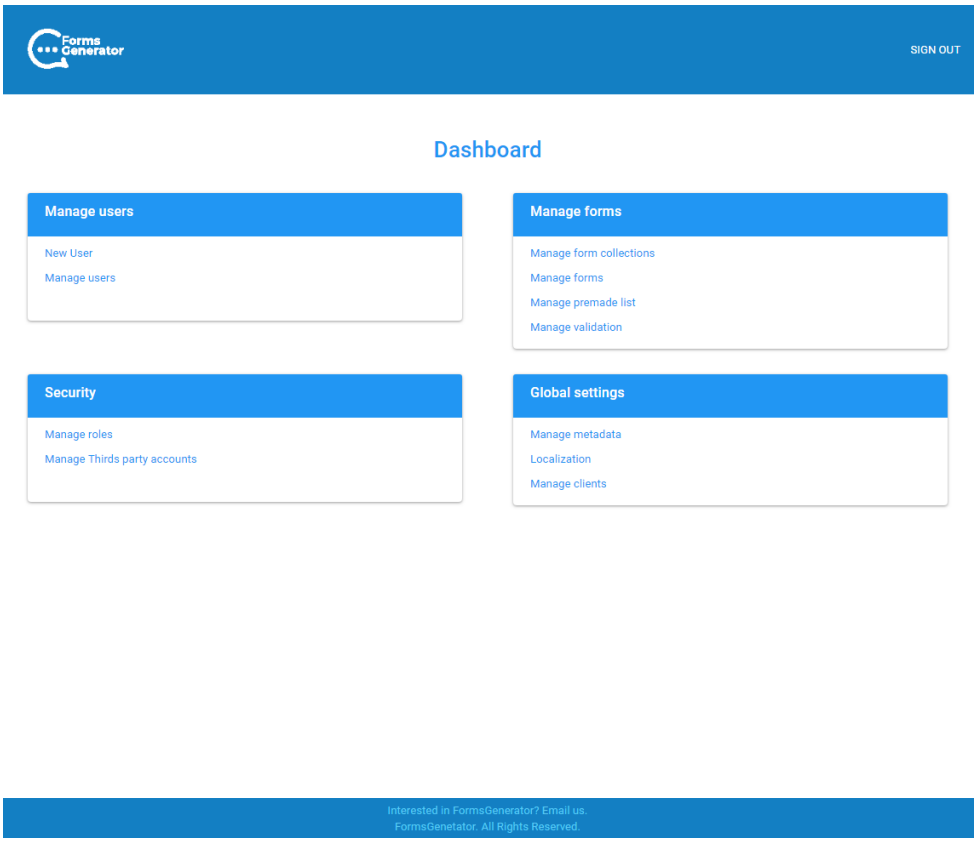

Рисунок 5.8 – Сторінка навігації по системі

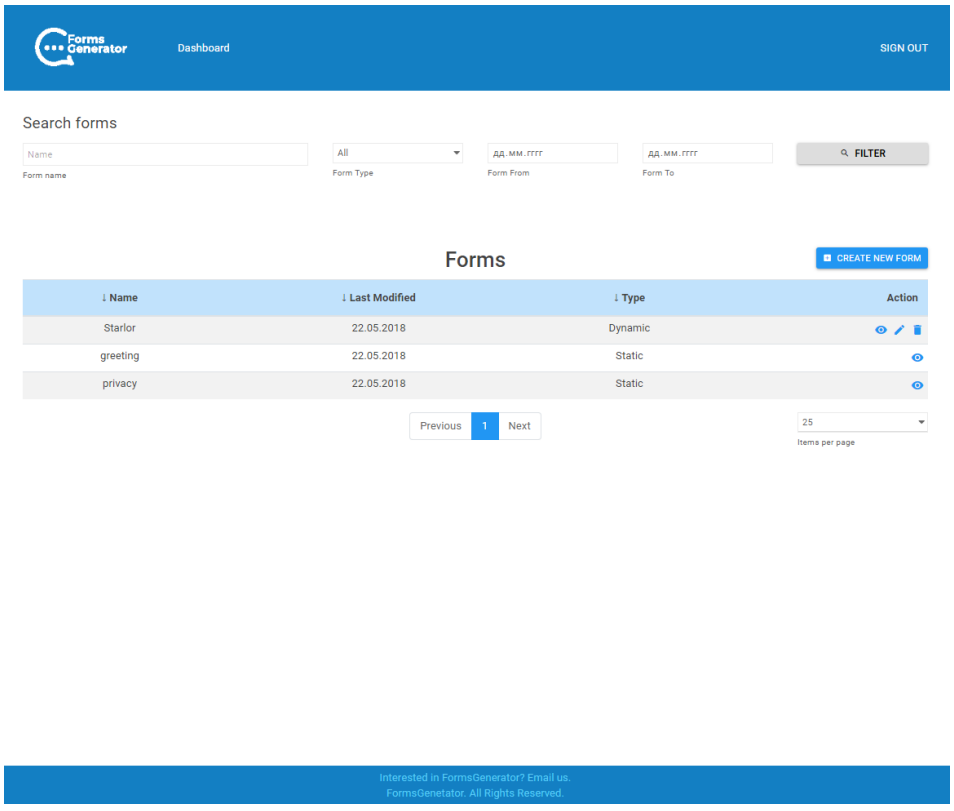

Рисунок 5.9 – Сторінка з списком форм

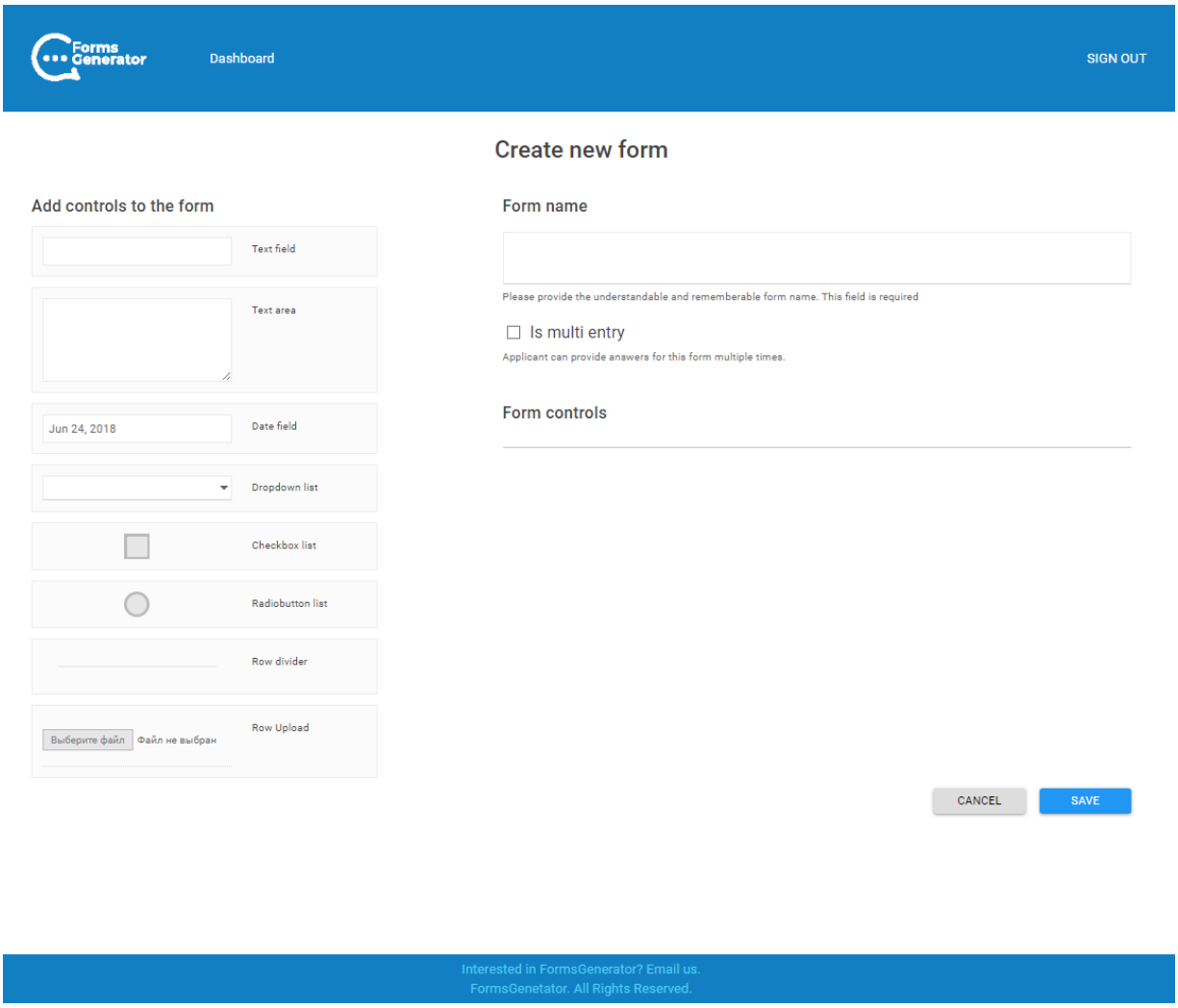

Рисунок 5.10 – Сторінка створення та редагування форм

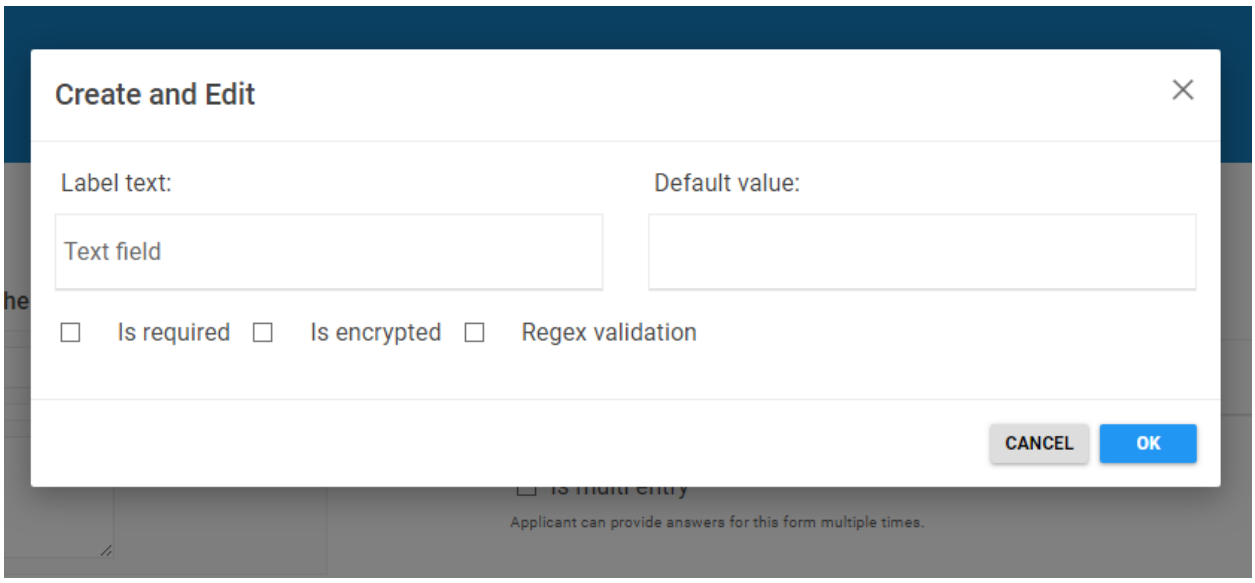

Рисунок 5.11 – Модальне вікно додавання та редагування Text field

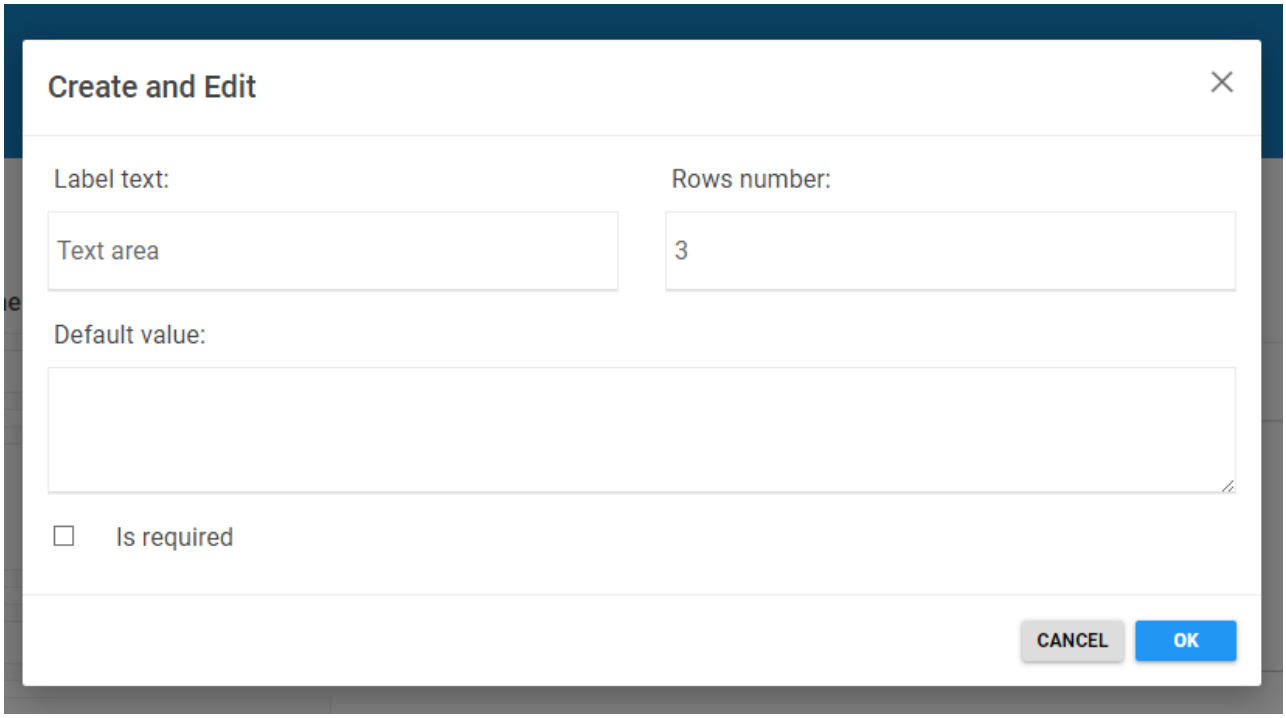

Рисунок 5.12 – Модальне вікно додавання та редагування Text area

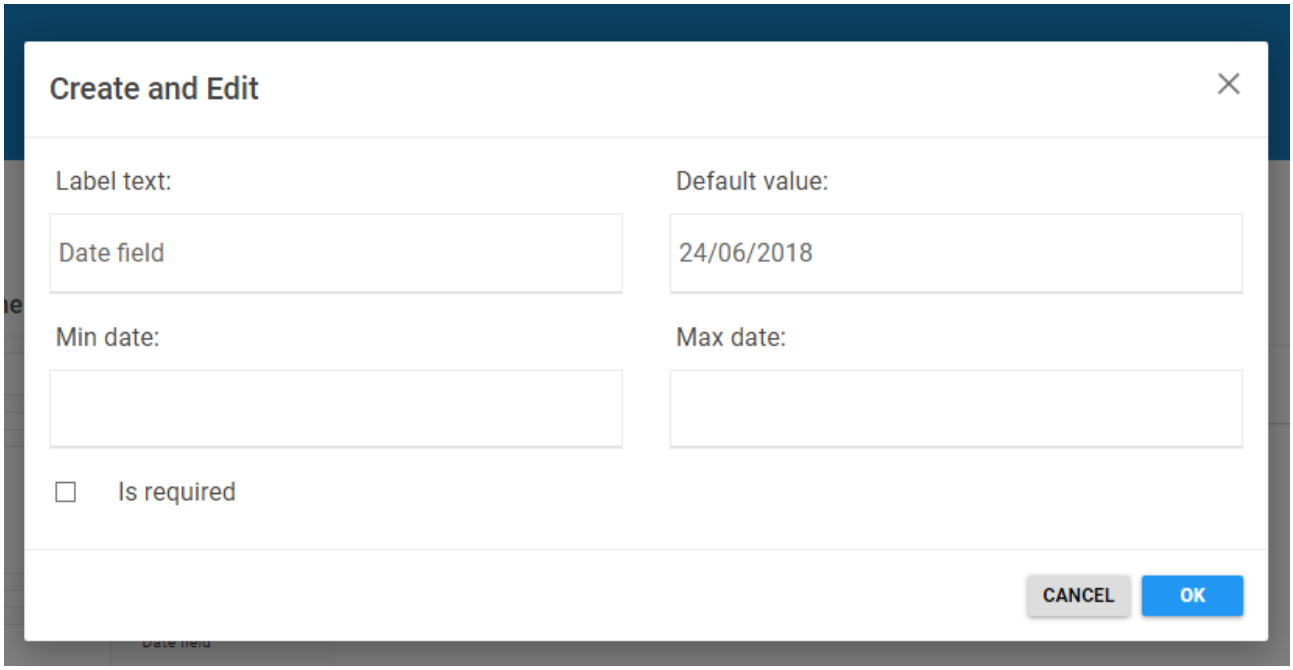

Рисунок 5.13 – Модальне вікно додавання та редагування Datepicker

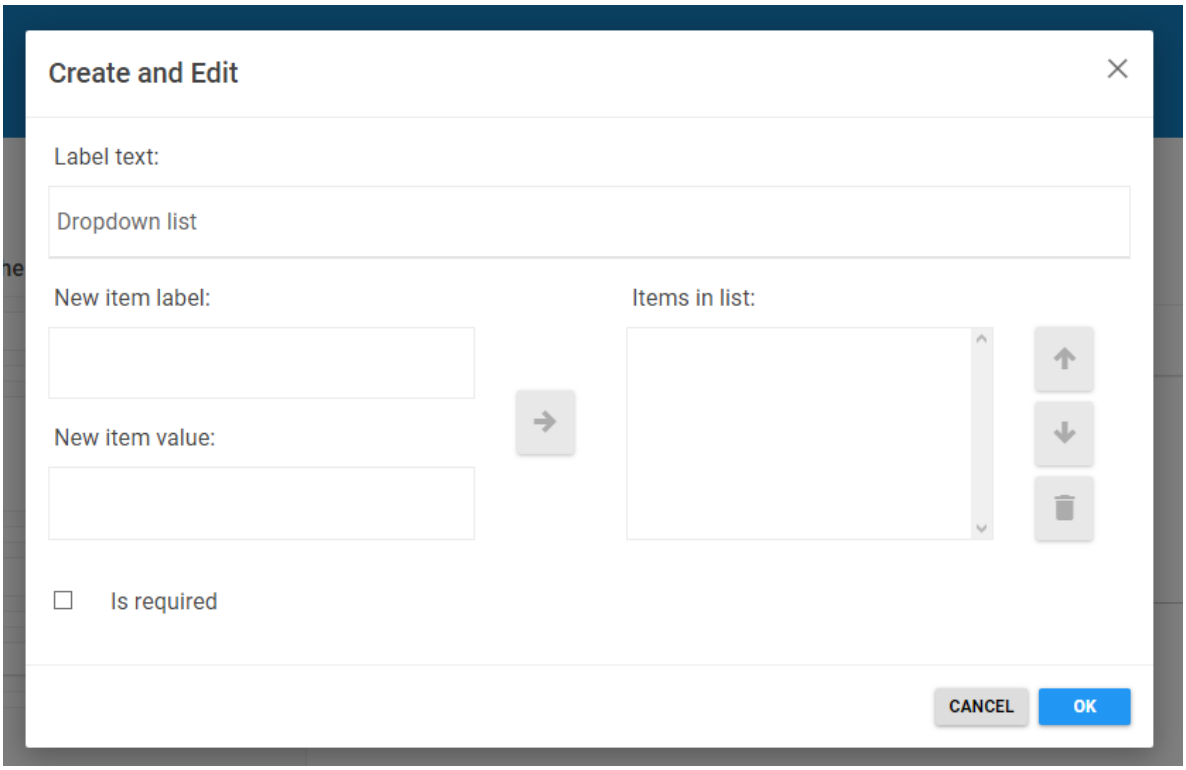

Рисунок 5.14 – Модальне вікно додавання та редагування Dropdown

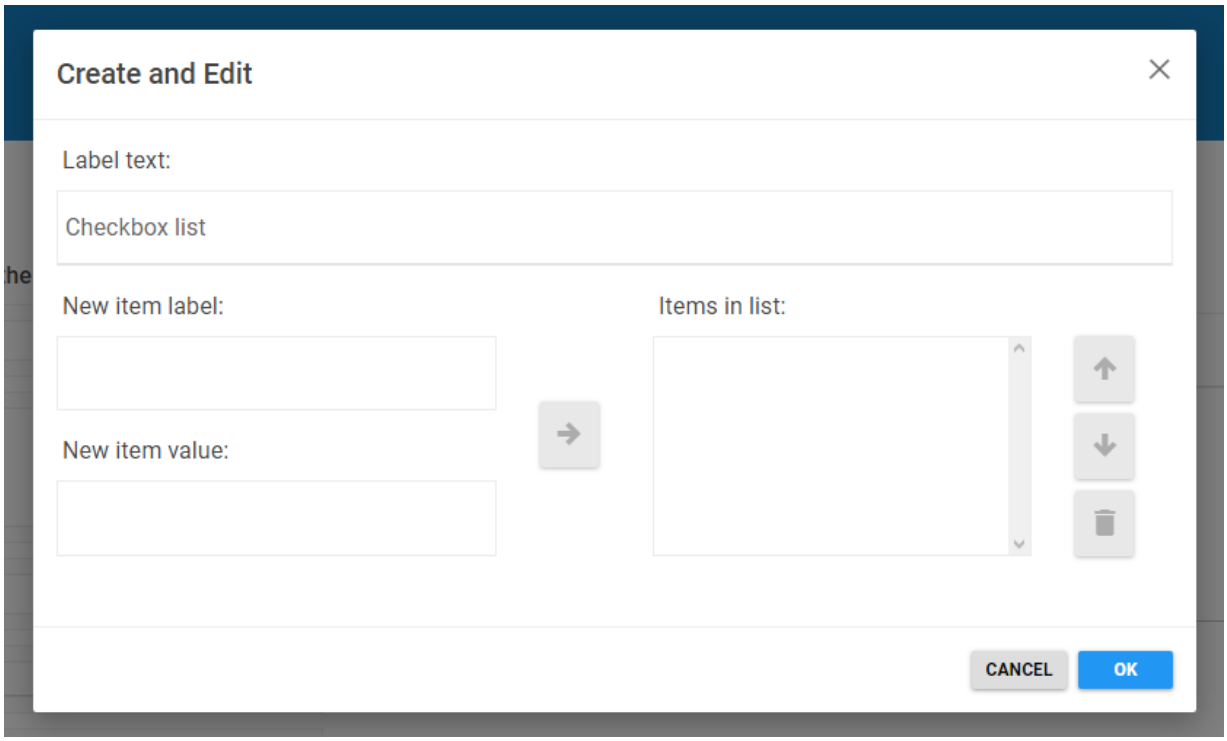

Рисунок 5.15 – Модальне вікно додавання та редагування Checkbox

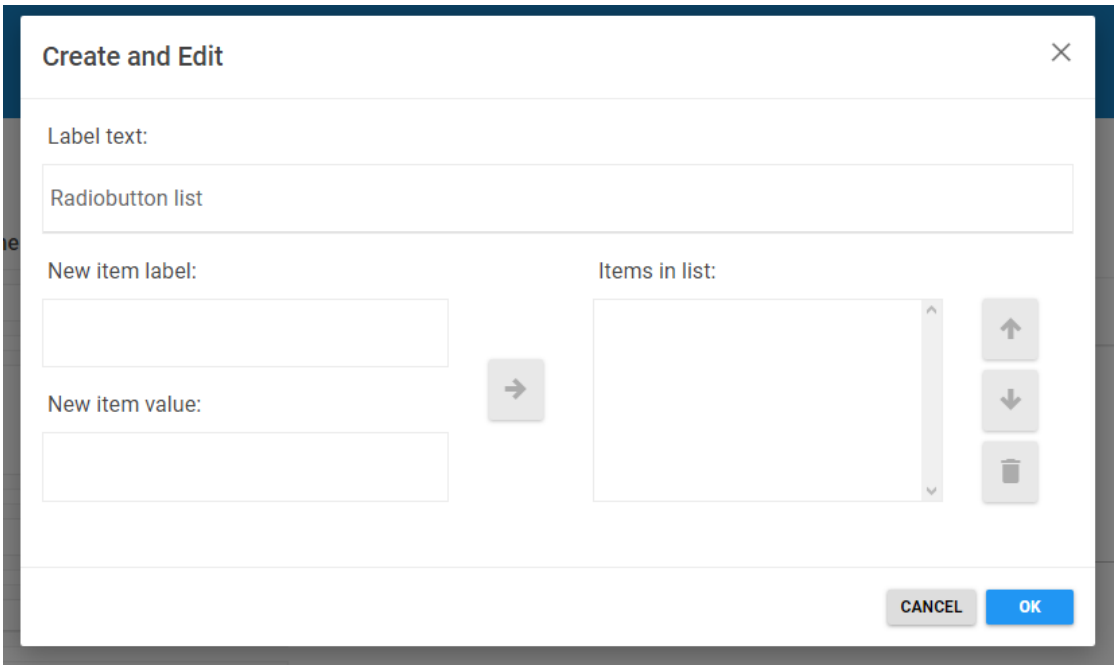

Рисунок 5.16 – Модальне вікно додавання та редагування Radiobutton

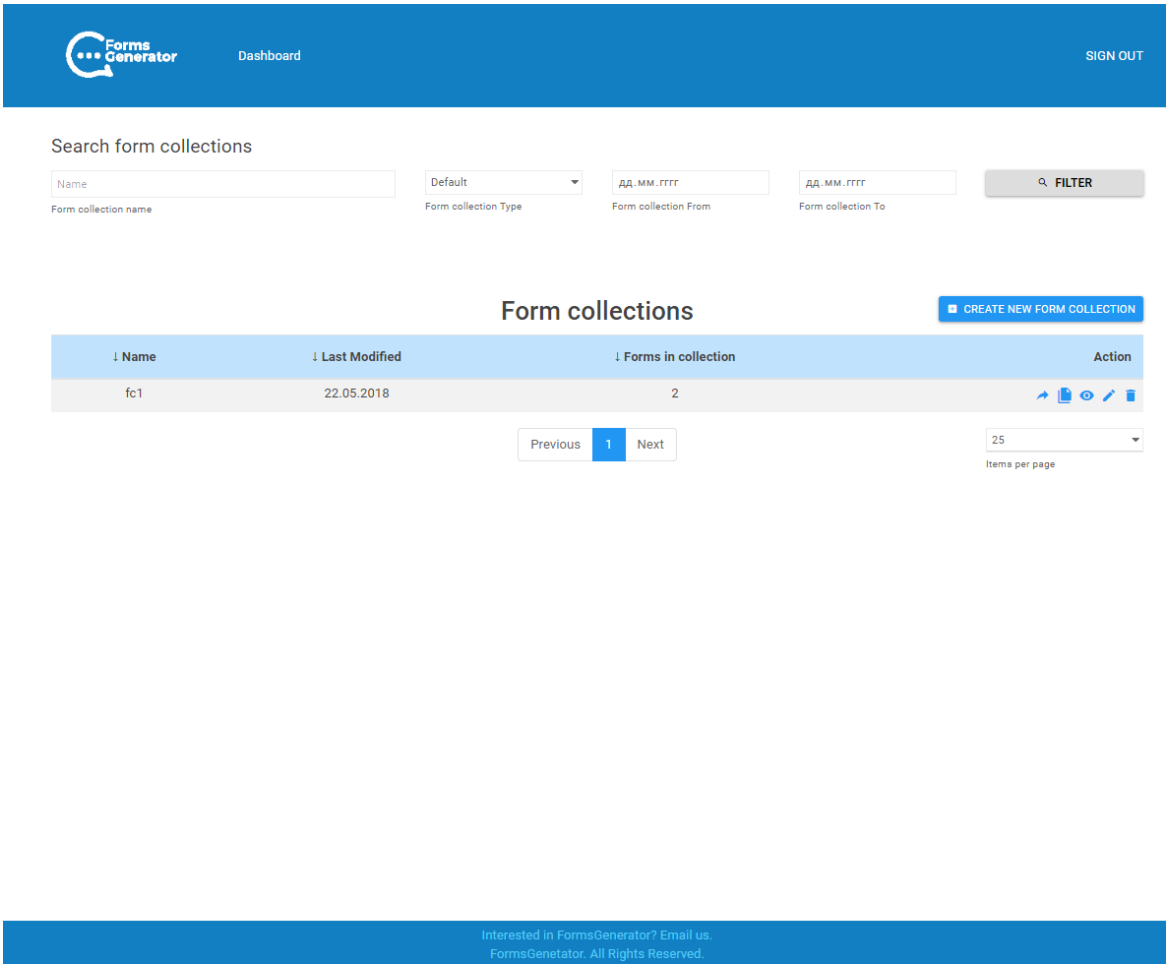

Рисунок 5.17 – Сторінка з списком колекцій форм

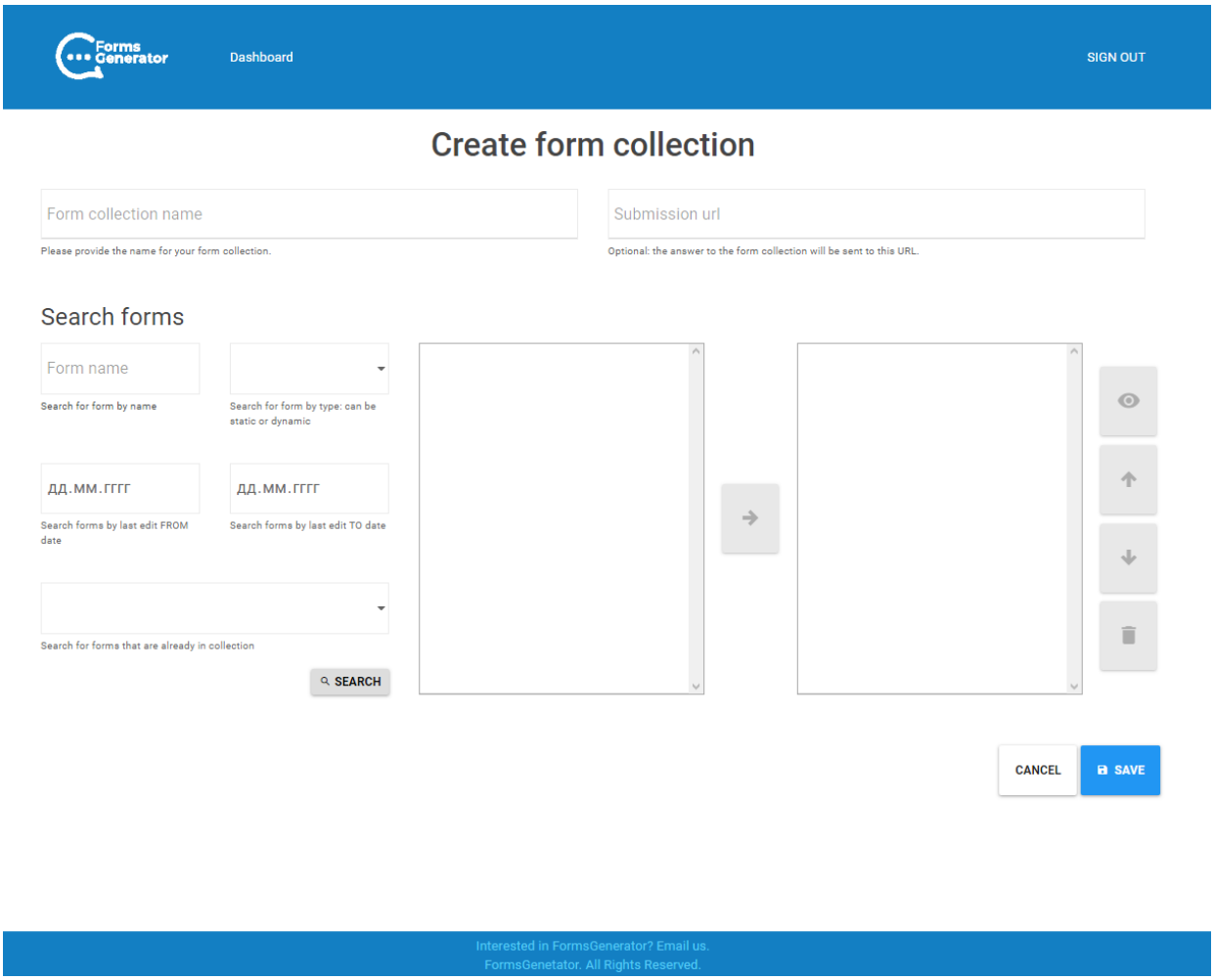

Рисунок 5.18 – Сторінка створення та редагування колекцій форм

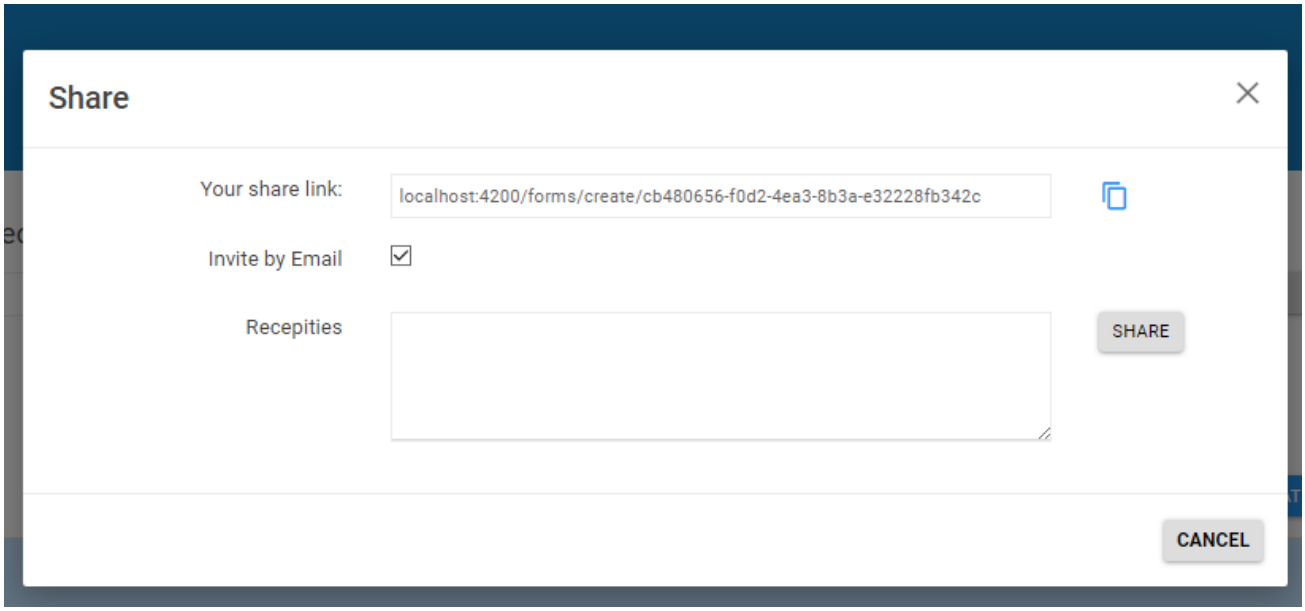

Рисунок 5.19 – Модальне вікно поширення колекцій форм

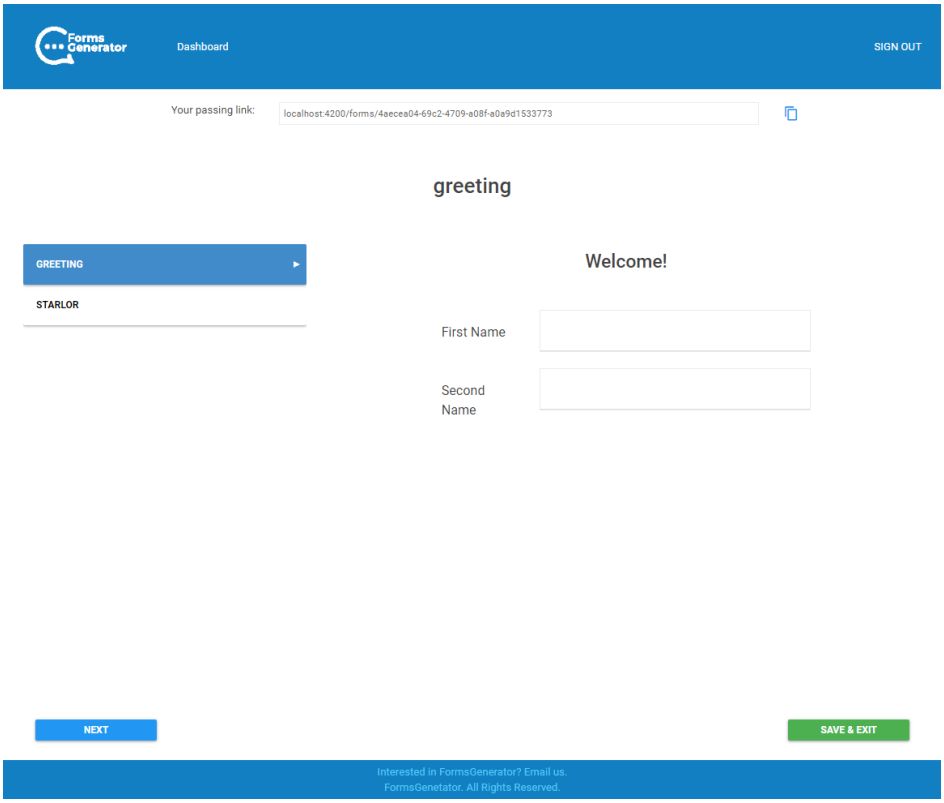

Рисунок 5.20 – Сторінка проходження колекції форм

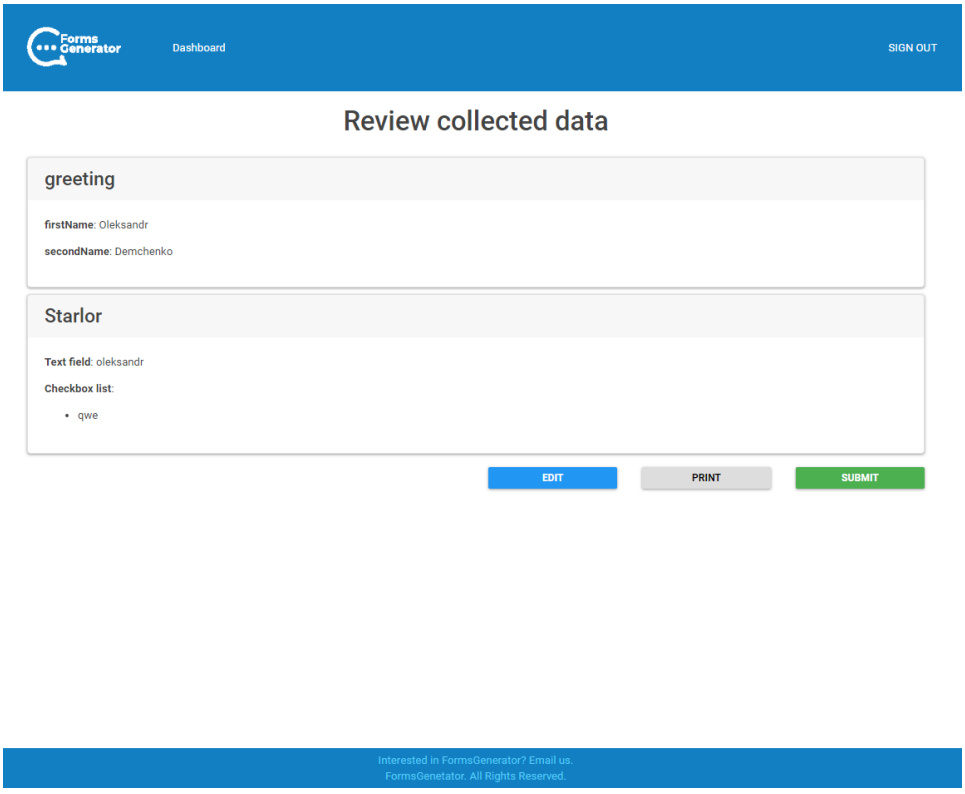

Рисунок 5.21 – Сторінка перегляду власних відповідей на колекцію форм

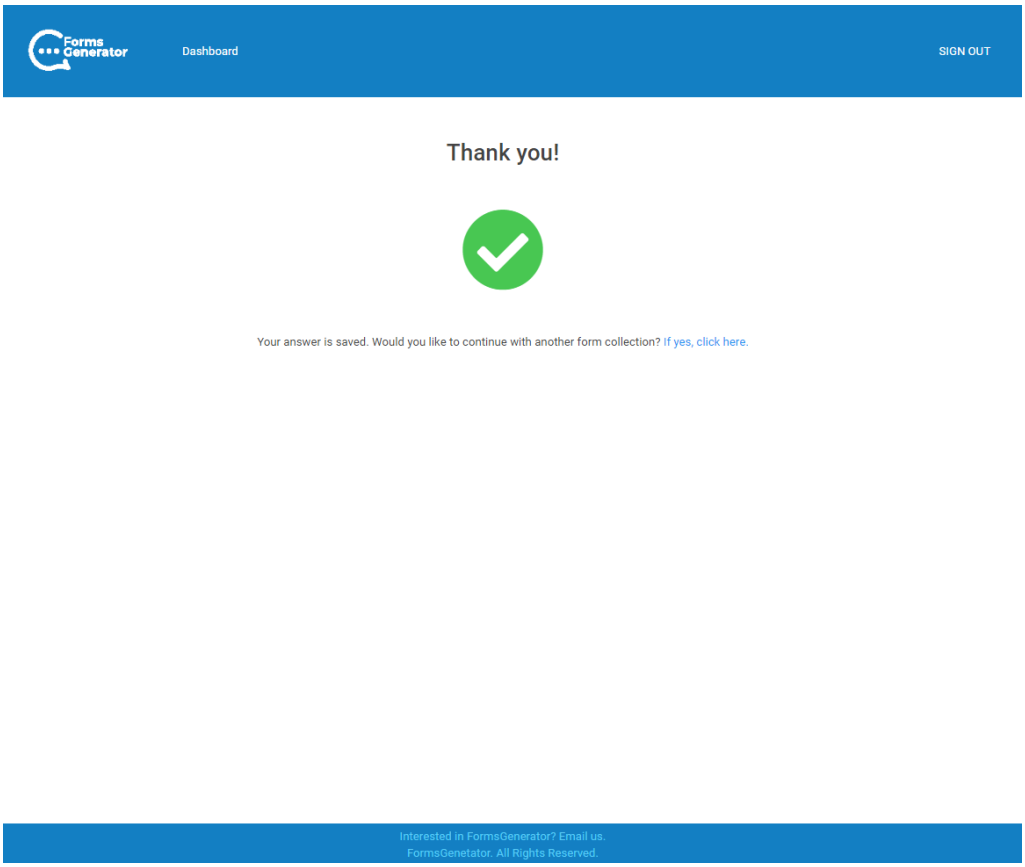

Рисунок 5.22 – Сторінка вдячності за проходження колекцій форм

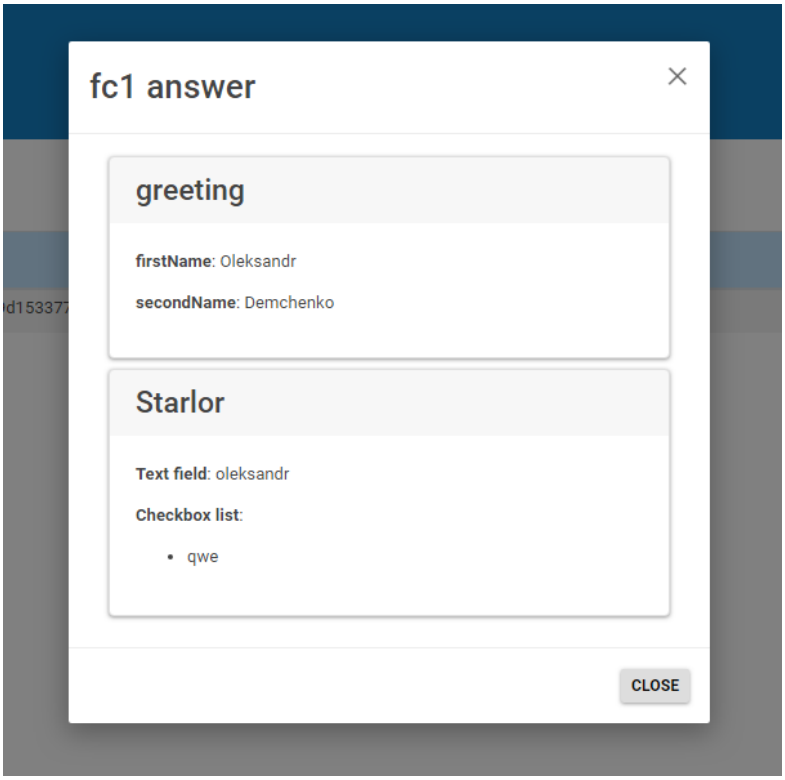

Рисунок 5.23 – Модальне вікно перегляду відповідей на колекцію форм

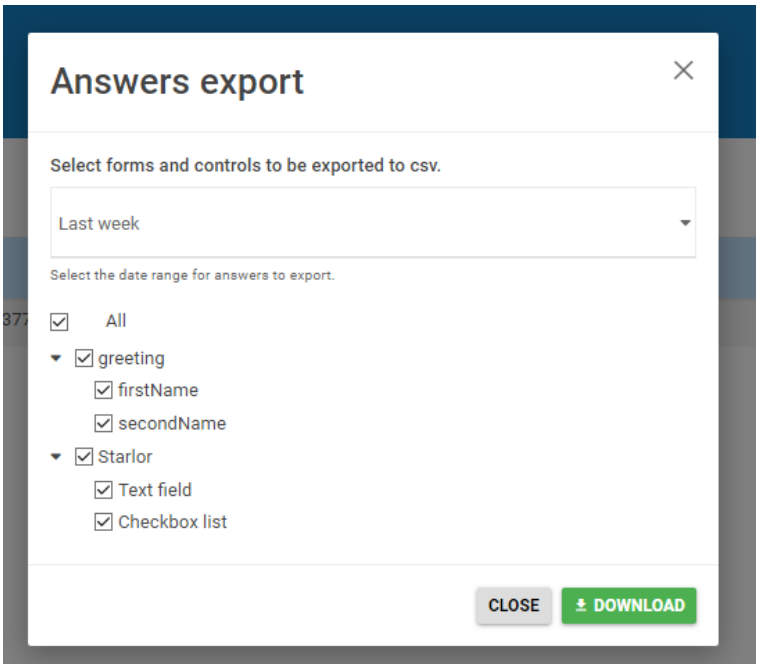

Рисунок 5.24 – Модальне вікно експорту колекції форм

5.9 Експерименти в обробці даних

Щоб перевірити ефективність Apriori, було проаналізувано 500 МБ, 1 ГБ, 2 ГБ, 4 ГБ і 8 ГБ інформації, отриманої з відповідей на колекції форм. Мінімальний поріг підтримки встановлено на 2%. Кожен набір вхідних даних виконувався щонайменше п’ять разів. Результати показують, що вихід алгоритма є послідовним, що підтверджувало стабільність і доступність Apriori.

Рисунок 5.25 – Час прогону алгоритма Apriori

Загальний час роботи алгоритма показано на рисунку 5.25. Використання процесора та використання пам'яті показано на рисунках 5.26 та 5.27.

Алгоритм Apriori завжди чекає повного сканування таблиці, використання процесора не є слабким місцем алгоритма. Але, алгоритм Apriori зберігає всі дані в пам’яті, при цьому збільшується використання оперативної памяті зі збільшенням набору даних. Якщо набір даних досить великий, алгоритм Apriori зіткнеться із вузькими місцями в пам'яті.

Рисунок 5.26 – Використання CPU алгоритмом Apriori

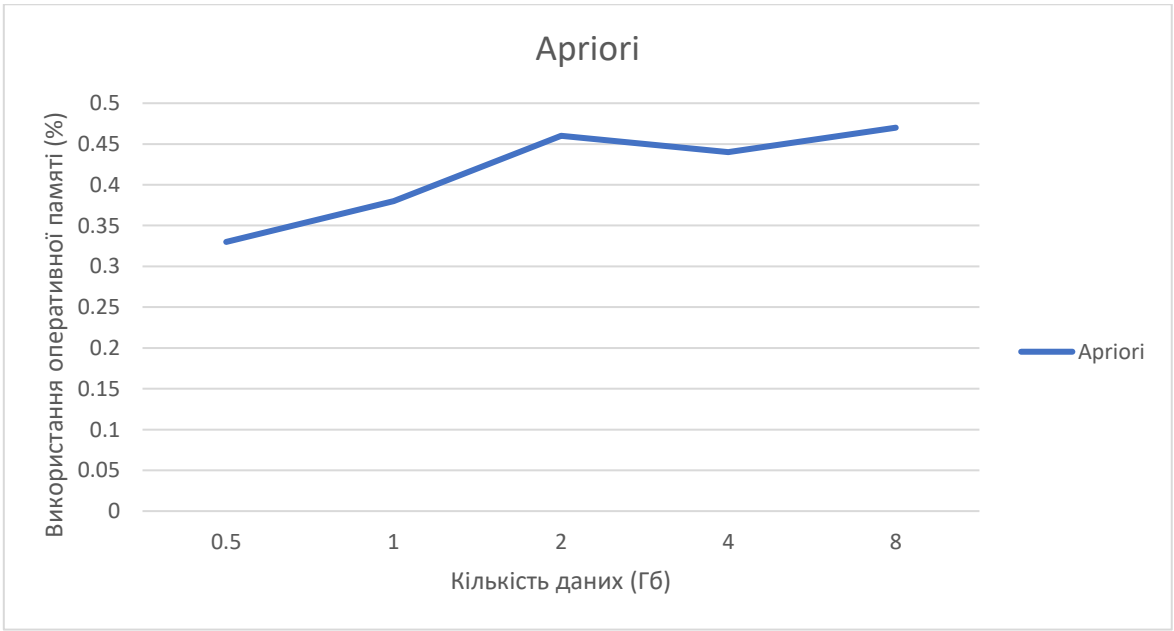

Рисунок 5.27 – Використання оперативної памяті алгоритмом Apriori

ВИСНОВКИ

У результаті виконання магістерської роботи було виконано огляд існуючих проблем збору та обробки даних, а також запропоновано новий метод обробки неструктурованих даних за допомогою Big Data та Data Mining.

На підготовчому етапі було проведено аналіз предметної галузі та її актуальність. Після порівняння вже існуючих систем, які схожі за тематикою, була сформована постановка задачі.

Далі було проведено долслідження методів пошуку асоціативних правил та був створений новий метод на основі метода Apriori, що дозволяє аналізувати неструктуранії дані.

В результаті було вирішено розробити прототип системи на основі нового методу пошуку асоціативних правил в неструктурованих даних та за допомогою технологій Big Data та Data Mining. На етапі проектування були розроблемі UML діаграми. Були визначені основні ролі користувачів та їх функціонал.

Також, перед початком розробки прототипа, були проаналізовані мови програмування, фреймворки та архітектури. Вибір був обґрунтованим.

На наступному етапі новий метод було реалізовано в прототипі програмного засобу за допомогою технологій ASP.NET Core, ASP.NET Identity, MongoDB та Angular, а також Big Data та Data Mining, а також за допомогою сучасних cloud-base технологій Azure. Були розроблені інтерфейси, сервіси, моделі представлення, самі сутності і репозиторії для даного проекту. Також використовувалися такі відомі шаблони, як Respository та Unit of Work.

Завершалним етапом стало проведення тестових експериментів на створеній системі з наборами даних різного розміру (500 Мб, 1 Гб, 2 Гб, 4 Гб та 8 Гб).

Загалом, поставлена задача була успішно виконана в результаті виконнання роботи. Однак, в подальшому можливе проведення оптимізації нового методу пошуку асоціативних правил та реалізація нових методів Big Data та Data Mining для подальшої обробки даних та розширенню функціонала розробленої системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Новые медиа: социальная теория и методология исследований: словарь-справочник /отв.ред. О. В. Сергеева, О. В. Терещенко. — СПб.: Алетейя, 2015. — 264 с. — ISBN 978-5-906792-46-4
2. Sproull, L. and Kiesler, S. (1986) ‘Reducing social context cues: electronic mail in organizaitonal communication’, Management Science, 32(11): 1492-1512.
3. Anderson, S. E., & Gansneder, B. M. (1995). Using electronic mail surveys and computer-monitored data for studying computer-mediated communication systems. Social Science Computer Review, 13(1), 33–46.
4. Using electronic mail (e-mail) surveys for geographic research: Lessons from a survey of Russian environmentalists. SRM O'Lear – The Professional Geographer, 1996
5. Kehoe, C. M., & Pitkow, J. E. (1996, Summer). Surveying the territory: GVU’s five WWW user surveys. World Wide Web Journal
6. Klaus M. Schmidt. The Review of Economic Studies, Volume 64, Issue 2, April 1997
7. T.M.F. Smith. (1997) Social surveys and social science. The Canadian Journal of Statistics, 25, 23-44.
8. Sheehan, B. K., & McMillan, J. S. (1999). Response variation in e-mail surveys: An exploration. Journal of Advertising, 39(4), 45-54.
9. Richard J. Harris (1997). Significance Tests Have Their Place, University of New Mexico
10. Dillman, D. A. (2000). Mail and Internet Surveys: The Tailored Design Method (2nd ed.). New York: Wiley 464 pp.
11. Mick P. Couper (2000). Review: Web Surveys: A Review of Issues and Approaches. Public Opinion Quarterly, Volume 64, Issue 4, February 2000, Pages 464–494
12. Schuldt, B.A. and Totten, J.W. (1994), ‘Electronic mail vs. mail survey response rates’, Marketing Research, 6(1): 36-9.
13. Comley P. (1996) The Use of the Internet as a Data Collection Method
14. Филиппова Т. В. Интернет как инструмент социологического исследования//Социологические исследования. 2001. № 9. С. 115—122. http://ecsocman.hse.ru/data/165/675/1216/020Filippova.pdf
15. Докторов Б. З. Онлайновые опросы: обыденность наступившего столетия // Телескоп: наблюдения за повседневной жизнью петербуржцев, 2000. № 4

16. Mario Callegaro, Reg Baker, Jelke Bethlehem, Anja S. Göritz, Jon A. Krosnick and Paul J. Lavrakas. Online panel research. History, concepts, applications and a look at the future//Online Panel Research: A Data Quality Perspective, Wiley (2014), pp. 1-22
17. Fielding N. G., Lee R. M., Blank G. (ed.). The SAGE handbook of online research methods. — Sage, 2008
18. Офіційний сайт SurveyMonkey. Режим доступу [URL]: https://www.surveymonkey.ru
19. Офіційний сайт Google з описом можливостей Google Forms. Режим доступу [URL]: https://www.google.com/intl/ru_ua/forms/about/
20. Офіційний сайт TypeForm. Режим доступу [URL]: https://www.typeform.com
21. F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber, “Bigtable: A Distributed Storage System for Structured Data Research,” Google, 2006.
22. S. Ghemawat, H. Gobioff, and S. Leung, “The Google file sys-tem,” Proceedings of the nineteenth ACM symposium on Operat-ing systems principles, Vol. 37, Issue 5, pp. 29-43, 2003. http://dx.doi.org/10.1145/945445.945450
23. J. Dean, and S. Ghemawat, “MapReduce: Simplified data pro-cessing on large clusters,” Proceedings of the 6th OSDI, pp. 137-150, 2004.
24. Буч, Г. Язык UML / Г. Буч – М.: ДМК Пресс, 2006. – 248 с.
25. Рихтер Дж. – CLR via C# / Redmond, Washington 98052-6399
26. Troelsen A. Pro C# 7 With .NET and .NET Core / A. Troelsen, P. Japikse.
27. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов. – 4-е изд. [Текст]: справочник / М. Мак-Дональд, А. Фримен, М. Шпушта – М.: ООО "И.Д. Вильямс", 2011. – 1424 с.
28. Троелсен Е. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание / Е. Троелсен – М.: «Вильямс», 2013. – 1312 с.
29. Overview of ASP.NET Core MVC [Електронний ресурс] – Режим доступу до ресурсу:https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.0
30. Reenskaug T. The DCI Architecture: A New Vision of Object-Oriented Programming [Електронний ресурс] / T. Reenskaug, J. O. Coplien. – 2009. – Режим доступу до ресурсу: https://www.artima.com/articles/dci_vision.html
31. REST API Tutorial [Електронний ресурс] / REST API Tutorial. – Режим доступу: https://www.restapitutorial.com/ – Назва з екрана.

32. Голощапов А. REST программирование для приложений / А. Голощапов – БХВ−Петербург, 2011. – 438с.
33. ASP.NET Web API [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/previous-versions/aspnet/web-frameworks/hh833994(v=vs.108)
34. What is Angular? [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs
35. F. Diebold, “Big Data: Dynamic Factor Models for Macroeconomic Measurement and Forecasting,” Discussion read to the Eighth World Congress of the Econometric Society, 2000.
36. D. Laney, “3-D Data Management: Controlling Data Volume, Velocity and Variety”, February 6, 2018.
37. G. Widmer, and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” Journal of Machine Learning, Vol. 23, Issue 1, pp. 69-101, 1996.
38. J. Gama, A. Bifet, I. Zliobaite, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation,” ACM Computing Surveys, Vol. 1, No. 1, 2017.
39. T. Zhang, R. Ramakrishnan, and M. Livn, “BIRCH: A new data clustering algorithm and its applications,” Data Mining and Knowledge Discovery, Vol. 1, Issue 2, pp. 141-182, 1997.
40. 24-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. – Харків: ХНУРЕ. 2020
41. P. S. Bradley, U. M. Fayyad, and C. Reina, “Scaling clustering algorithms to large databases,” Proceedings of Knowledge Discovery and Data Mining, AAAI Press, 1998.
42. F. Farnstrom, J. Lewis, and C. Elkan, “Scalability for clustering algorithms revisited,” SIGKDD Explorations, 2018.
43. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, “Clustering data streams,” IEEE Symposium on Foundations of Computer Science, 2016.
44. L. O’Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, “Streaming data algorithms for high-quality clustering,” 2017.
45. C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams”, 2019.
46. P. Rodrigues, J. Gama, and J. Pedroso, “ODAC: Hierarchical clustering of time series data streams,” Proceedings of the Sixth SIAM International Conference on Data Mining, 2006.

47. F. Cao, M. Ester, W. Qian, and A. Zhou, “Density-based cluster-ing over an evolving data stream with noise,” Proceedings of the Sixth SIAM International Conference on Data Mining, SIAM, 2016.
48. Y. Chen, and L. Tu, “Density-based clustering for real-time stream data,” Proceedings of the 13th ACM SIGKDD international con-ference on Knowledge discovery and data mining, ACM Press, 2007.
49. A. Zhou, F. Cao, W. Qian, and C. Jin, “Tracking clusters in evolv-ing data streams over sliding windows,” Knowledge and Infor-mation Systems, 2018.
50. P. Kranen, I. Assent, C. Baldauf, and T. Seidl, “The clustree: indexing micro-clusters for anytime stream mining,” Knowledge and Information Systems, Vol. 29, No. 2, 2011.
51. J. Gama, P. P. Rodrigues, and L. Lopes, “Clustering distributed sensor data streams using local processing and reduced communi-cation,” Intelligent Data Analysis, Vol. 15, No. 1, 2017.
52. M. R. Ackermann, M. Martens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, “Streamkm++: A clustering algorithm for data streams,” ACM Journal of Experimental Algorithmics, Vol. 17, 2018.
53. H. He, and H. Man, “SOMKE: Kernel Density Estimation Over Data Streams by Sequences of Self-Organizing Maps,” IEEE Transactions on Neural Networks and Learning Systems, Vol. 23, No. 8, 2012.
54. J. Roure, and R. Sanguesa, “Incremental Methods for Bayesian Network Learning,” 1999.
55. N. A. Syed, H. Liu, and K. K. Sung, “Handling concept drift in incremental learning with support vector machines,” Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999.
56. P. Domingos, and G. Hulten, “Mining high-speed data streams,” Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.
57. G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.
58. W. N. Street, and Y. A. Kim, “Streaming Ensemble Algorithm (SEA) for large-scale classification,” Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

59. H. Wang, W. Fan, P.S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013.
60. J. Z. Kolter, and M. A. Maloof, “Dynamic Weighted Majority: A New Ensemble Method for Drifting Concepts,” ACM Journal of Machine Learning Research, Vol. 8, 2017.
61. E. Ikonomovska, J. Gama, R. Sebastiao, and D. Gjorgjevik, “Re-gression Trees from Data Streams with Drift Detection,” Proceedings of the International Conference on Data Science, 2019.
62. J. H. Chang, and W. S. Lee, “Finding Recent Frequent Itemsets Adaptively over Online Data Streams,” Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, , 2013.
63. C. Gianella, J. Han, J. Pei, X. Yan, and P. S. Yu, “Mining Frequent Patterns in Data Streams at Multiple Time Granularities,” Data Mining: next generation challenges and future directions, MIT/AAAI Press, 2014.
64. Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, “Moment: Maintain-ing Closed Frequent Itemsets over a Stream Sliding Window,” 4th IEEE International Conference on Data Mining, 2014.
65. J. Cheng, Y. Ke, and W. Ng, “Maintaining frequent closed item-sets over a sliding window,” Journal of Intelligent Information Systems, 2018.
66. S. J. Yen, C. W. Wu, Y. S. Lee, V. S. Tseng, and C. H. Hsieh, “A Fast Algorithm for Mining Frequent Closed Itemsets over Stream Sliding Window,” IEEE International Conference on Fuzzy Systems, 2011.
67. D. Yang, E. Rundensteiner, and M. Ward, “Neighbor-based pat-tern detection for windows over streaming data,” In EDBT, 2019.
68. F. Angiulli, and F. Fassetti, “Distance-based outlier queries in data streams: the novel task and algorithms,” Data Mining and Knowledge Discovery, 2010.
69. D. Georgiadis, M. Kontaki, A. Gounaris, A. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, “Continuous Outlier Detection in Data Streams: An Extensible Framework and State-Of-The-Art Algorithms,” SIGMOD, 2013.
70. G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk, “Scalable Collabo-rative Filtering Approaches for Large Recommender Systems,” Journal of Machine Learning Research, 2019.

71. S. Papadimitriou, J. Sun, and C. Faloutsos, “Streaming Pattern Discovery in Multiple Time-Series,” Proceedings of 31st VLDB Conference, 2016.
72. Z. F. Siddiqui, and M. Spiliopoulou, “Combining Multiple Interrelated Streams for Incremental Clustering,” SSDBM, 2019.
73. Z. F. Siddiqui, and M. Spiliopoulou, “Tree Induction over Perennial Objects,” SSDBM, 2010.
74. E. Ikonomovska, “Regression on evolving multi-relational data streams,” LEMIR, 2017.
75. D. Wegener, M. Mock, D. Adranale, and S. Wrobel, “Toolkit-based High-Performance Data Mining of Large Data on MapReduce Clusters,” Proc. Int’l Conf. Data Mining Workshops (ICDMW’09), 2009.
76. S. Das, Y. Sismanis, K. S. Beyer, R. Gemulla, P. J. Haas, and J. McPherson, “Ricardo: Integrating R and Hadoop,” Proceedings ACM SIGMOD Int’l Conf. Management Data, 2019.
77. C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun, “Map-Reduce for Machine Learning on Multi-core,” Proc. 20th Ann. Conf. Neural Information Processing Systems, 2016.