КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка та програмна реалізація багатокритеріальної оцінки та відбору підприємств для кредитування»

Студента 4 курсу, 9 групи,
спеціальності
122 «Комп'ютерні науки» _____ Грімова Артема Ігоровича
(підпис студента)

Науковий керівник:
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
(підпис керівника)

Гарант освітньої програми:
кандидат технічних наук, доцент _____ Демідов Павло Георгійович
(підпис керівника)

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем

Спеціальність 122 «Комп'ютерні науки»

Затверджую

Зав. кафедри _____ Пурський О.І.

«20» грудня 2019 р.

ЗАВДАННЯ

на випускний кваліфікаційний проект студенту

Грімову Артему Ігоровичу

(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту (далі ВКП):

«Розробка та програмна реалізація багатокритеріальної оцінки та відбору підприємств для кредитування»

Затверджена наказом ректора від «04» грудня 2019 р. № 4111

2. Строк здачі студентом закінченого проекту «12» червня 2020 року

3. Цільова установка та вихідні дані до роботи

Мета роботи: дослідження методів та розробка програмного забезпечення оцінки та відбору підприємств для кредитування.

Об'єкт дослідження: процеси оцінки банком кандидатів на кредитування.

Предмет дослідження: моделі, методи та інформаційні технології в системі оцінювання та відбору підприємств для кредитування.

4. Перелік графічного матеріалу_____

_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Демідов П.Г.	15.12.2019 р.	15.12.2019 р.
2	Демідов П.Г.	15.12.2019 р.	15.12.2019 р.
3	Демідов П.Г.	15.12.2019 р.	15.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНО-МЕТОДОЛОГІЧНІ ОСНОВИ РЕАЛІЗАЦІЇ БАГАТОКРИТЕРІАЛЬНОЇ ОЦІНКИ ТА ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ

1.1. Огляд проблематики кредитування підприємств фінансовими установами

1.2. Методи прийняття рішень на основі теорії нечітких множин

1.3. Характеристики систем управління банківською діяльністю та місце задач кредитування у їх складі

РОЗДІЛ 2. ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК)

2.1. Постановка задачі

2.2. Метод максимінної згортки (з урахуванням однакової ваги критеріїв) у сфері банківського кредитування

2.3. Аналітичний апарат розрахунку коефіцієнтів кредитоспроможності позичальників

2.4. Методи побудови функції приналежності. Лінійна інтерполяція

РОЗДІЛ 3. РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК)

3.1. Загальна характеристика КСВПК

3.2. Алгоритмічна реалізація методів у сфері банківського кредитування

3.3. Програмна реалізація КСВПК

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№	Назва етапів ВКП	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми ВКП	01.10.2019	01.10.2019
2	Розробка та затвердження завдання на ВКП	15.12.2019	15.12.2019
3	Вступ	03.02.2020	03.02.2020
4	Розділ 1. Теоретично-методологічні основи реалізації багатокритеріальної оцінки та відбору підприємств для кредитування	28.02.2020	28.02.2020
5	Розділ 2. Проектування комп'ютерної системи відбору підприємств для кредитування (КСВПК)	06.04.2020	06.04.2020
6	Розділ 3. Розробка комп'ютерної системи відбору підприємств для кредитування (КСВПК)	12.05.2020	12.05.2020
7	Висновки	15.05.2020	15.05.2020
8	Здача ВКП на кафедру науковому керівнику	20.05.2020	20.05.2020
9	Попередній захист ВКП	03.06.2020	03.06.2020
10	Виправлення зауважень, зовнішнє рецензування ВКП	09.06.2020	09.06.2020
11	Представлення готового зшитого ВКП на кафедру	12.06.2020	12.06.2020
12	Публічний захист ВКП	За розкладом роботи ЕК	За розкладом роботи ЕК

8. Дата видачі завдання « 15 » грудня 2019 р.

Керівник випускного кваліфікаційного проекту

Демідов П.Г.

(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.

(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Грімов А.І.

(прізвище, ініціали, підпис)

9. Відгук керівника випускного кваліфікаційного проекту

_____

Керівник випускного кваліфікаційного проекту

_____ « » 2020 р.
(підпис)

10. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента Грімова А.І.
(прізвище, ініціали)

може бути допущений до захисту в екзаменаційній комісії.

Гарант освітньої програми _____ Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2020 р.

Анотація

У випускному кваліфікаційному проекті проведено аналіз методів оцінки та відбору підприємств для кредитування, теоретично обґрунтовано актуальність та її практичну значимість і розроблено програмне забезпечення, що дає змогу вирішити поставлене завдання на базі коефіцієнтів кредитоспроможності позичальників та теорії нечітких множин. В результаті отримано веб-сайт, який дає змогу здійснити усі необхідні розрахунки та прийняти правильне рішення.

Ключові слова: багатокритеріальна оцінка, банк, підприємство, кредит, нечіткі множини, функція приналежності.

Annotation

The analyze of the evaluation and selection methods of enterprises for lending is done in the graduation qualification project, the relevance and its practical significance is theoretically substantiated and software that allows to solve the problem based on the coefficients of creditworthiness of borrowers and fuzzy set theory is developed. As the result a website that allows you to make all the necessary calculations and to make the right decision is got.

Key words: multicriteria evaluation, bank, enterprise, credit, fuzzy sets, membership function.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. ТЕОРЕТИЧНО-МЕТОДОЛОГІЧНІ ОСНОВИ РЕАЛІЗАЦІЇ БАГАТОКРИТЕРІАЛЬНОЇ ОЦІНКИ ТА ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ 10
1.1. Огляд проблематики кредитування підприємств фінансовими установами 10
1.2. Методи прийняття рішень на основі теорії нечітких множин 14
1.3. Характеристики систем управління банківською діяльністю та місце задач кредитування у їх складі 23
РОЗДІЛ 2. ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК) 26
2.1. Постановка задачі 26
2.2. Метод максимінної згортки (з урахуванням однакової ваги критеріїв) у сфері банківського кредитування 26
2.3. Аналітичний апарат розрахунку коефіцієнтів кредитоспроможності позичальників 28
2.4. Методи побудови функції приналежності. Лінійна інтерполяція 30
РОЗДІЛ 3. РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК) 35
3.1. Загальна характеристика КСВПК 35
3.2. Алгоритмічна реалізація методів у сфері банківського кредитування 39
3.3. Програмна реалізація КСВПК 42
ВИСНОВКИ 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 48
ДОДАТКИ 52

ВСТУП

Стимулювати економіку країни, зробити її функціонування постійним, забезпечити стійке зростання – це завдання, виконання якого неможливе без ефективної банківської системи. Сучасні вітчизняні банки сприяють розвитку економіки за допомогою їх пріоритетної функції – кредитування, що дозволяє забезпечити швидкий і цивілізований розвиток підприємницької діяльності на внутрішньому і зовнішньому економічному просторі. Світовий досвід свідчить, що ринкова економіка оптимально об'єднує діяльність великих, середніх і малих підприємств. Незважаючи на те, що малий і середній бізнес є найбільш важливим елементом економіки будь-якої держави, він, в той же час, найбільш чутливий до будь-яких економічних і політичних змін. У зв'язку з тим, що організація взаємин малого і середнього бізнесу з банками є одним з найважливіших чинників розвитку цього сектора економіки, потреба в детальному дослідженні проблем і перспектив розвитку банківського кредитування є досить **актуальною**.

Питаннями, пов’язаними з особливостями організації кредитування українськими банками підприємств малого і середнього бізнесу, присвячені наукові праці таких вчених, як З. Варналія [7], І. Волошина [9], С. Глущенко [10], С. Дмитрова [11], Р. Коцовской [17], Е. Карпова, Л. Чубарєвої [16], Д. Єдновицький [12] та ін.

Визнаючи важливість проведених наукових досліджень, слід зазначити, що автори цих робіт розглядають окремі аспекти багатогранної проблеми, пов’язаної з визначенням сутності, умов і особливостей розвитку малого і середнього бізнесу. Однак до сих пір залишається невирішеним питання організації взаємовідносин банківських установ і підприємств малого та середнього бізнесу з урахуванням їх взаємних інтересів. Саме тому ця проблема потребує додаткового дослідження, враховуючи стрімкий розвиток банківського сектора і новітні реалії української економіки в процесі реалізації ринкових відносин.

Метою роботи є дослідження методів та розробка програмного забезпечення оцінки та відбору підприємств для кредитування.

Об'єктом дослідження є процеси оцінки банком його кандидатів на кредитування, а **предмет дослідження –** моделі, методи та інформаційні технології в системі оцінювання та відбору підприємств для кредитування.

Проведення всебічного аналізу показників підприємств дозволяє банку ефективніше управляти кредитними ресурсами і отримувати прибуток. Тому додаток, який забезпечує виконання багатокритеріальної оцінки та здійснює відбір підприємств безумовно матиме **практичне значення.**

РОЗДІЛ 1.
ТЕОРЕТИЧНО-МЕТОДОЛОГІЧНІ ОСНОВИ РЕАЛІЗАЦІЇ БАГАТОКРИТЕРІАЛЬНОЇ ОЦІНКИ ТА ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ

1.1. Огляд проблематики кредитування підприємств фінансовими установами

В даний час найпоширенішим видом кредиту є банківський. За такого кредитування підприємство виступає тільки в ролі позичальника. Підприємства можуть отримувати різноманітні види кредитів та послуги кредитного характеру (рис. 1.1).

Рис. 1.1. Види кредитів та послуг кредитного характеру

Перш ніж отримати кредит, підприємство повинно пройти певну процедуру по представленню інформації про свою діяльність. Проте вимоги і підходи, що застосовуються до малих та середніх підприємств значно відрізняються від тих, що застосовуються до великих.

Класифікація кредитів, що надаються підприємствам.

Кредити, які можуть отримати підприємства, класифікуються за такими ознаками: порядком надання, формами та видами, кредиторами, терміном надання, метою використання, забезпечення.

Кредиторами підприємств можуть бути: держава, міжнародні фінансово-кредитні установи, банки і спеціалізовані фінансово-кредитні інститути, підприємства.

Форми і види кредитів представлено на рисунку 1.2 [10].

Рис. 1.2. Форми і види кредитів, що надаються підприємствам

Види кредитів:

- лізинговий –відносини між суб'єктами господарювання, які що виникають під час оренди майна (лізинг-кредит / майновий кредит);

- державний – кредитні чи економічні відносини між суб'єктами господарювання і державою;

- комерційний – кредитні та економічнівідносини, що виникають між окремими підприємствами;

- банківський – економічні відносини між позичальником і кредитором з приводу надання коштів банком підприємству на умовах матеріального забезпечення, зворотності, платності, терміновості (такий кредит надається суб'єктам господарювання усіх форм власності на умовах, що передбачені через кредитний договір).

Державний і банківський кредити надаються підприємствам у грошовій формі, а комерційний та лізинговий – у товарній. Державний та банківський кредити погашаються у грошовій формі, а комерційний – зазвичай у грошовій формі. У період становлення ринкових відносин можлива його оплата як у товарній, так і в змішаній формах (грошовій і товарній). Лізинговий кредит може погашатися в товарній, грошовій і змішаній формах.

Особливості банківського кредитування підприємств малого та середнього бізнесу.

Розвиток світової економіки переконливо доводить найважливішу роль малих і середніх підприємств в національній економіці. Вони сприяють боротьбі з безробіттям за рахунок створення нових робочих місць, формування нормального конкурентного середовища. Малі підприємства в більшій мірі здатні реагувати на коливання споживчого попиту, на зміну кон'юнктури ринку, тим самим надаючи економіці додаткову стабільність. В процесі своєї діяльності підприємствам малого і середнього бізнесу доводиться вирішувати складні проблеми, пов'язані з фінансуванням господарських операцій. Зарубіжний досвід свідчить, що при пошуку зовнішніх джерел фінансових ресурсів основну увагу малі і середні підприємств приділяють банківському кредиту.

Витрати, пов'язані з високим ступенем ризику по кредитуванню підприємств малого і середнього бізнесу, банки, в кінцевому рахунку, перекладають на позичальників за допомогою високої ставки за кредитом або вимогою надати забезпечення у великому обсязі. В тому випадку, коли в силу високого ступеня банківського ризику знижується доступність кредиту на прийнятних для підприємств умовах, у зарубіжних малих і середніх підприємств є можливість скористатися різними видами підтримки, а саме, послугами спеціалізованих установ – гарантійних інститутів, втручання яких забезпечує зниження ступеня банківського кредитного ризику, і кредитних кооперативів, що надають малим та середнім підприємствам фінансування на вигідніших умовах в порівнянні з банками.

По відношенню до малих і середніх підприємств банки займають вкрай обережну позицію, розглядаючи даний кредитний ринок як дуже ризикований [27]. Можна виділити чотири основні чинники, які призводять до підвищення ступеня ризику по кредитуванню:

- власні кошти даної категорії підприємств обмежені;
- відсутність якісної інформації про стан своєї діяльності [28];
- відсутність репутації;
- певна частина активів підприємства носить специфічний характер.

Отже, в умовах невизначеності висока ймовірність того, що в разі зміни економічної ситуації мале або середнє підприємство не зможе виконати свої зобов'язання за кредитним договором, при цьому специфічні активи у зв'язку зі складнощами, пов'язаними з їх реалізацією, не зможуть забезпечити їх виконання [24].

Особливості банківського кредитування крупних підприємств.

Одним з основних ознак розвитку економіки є збільшення обсягів кредитування суб'єктів корпоративного бізнесу. Для кредитних установ кредитування корпоративного бізнесу є одним з основних видів операцій, так як займає істотну частку в загальному обсязі активів і, як наслідок, є однією з найбільш дохідних статей діяльності. Для суб'єкту корпоративного бізнесу банківський кредит є одним з основних джерел інвестицій і сприяє безперервності і прискоренню відтворювального процесу, і в цілому зміцнює економічний потенціал підприємства.

Найбільший інтерес у банків представляє корпоративний бізнес, оскільки суб'єкт великого підприємництва є для банку не просто клієнтом, але і рівноправним партнером.

Особливості [19]:

- при кредитуванні корпоративного бізнесу у вигляді овердрафту ліміт кредитування істотно вищий, ніж при роздрібному кредитуванні;
- кредитування підприємств корпоративного бізнесу завжди здійснюється безготівковим шляхом;
- при кредитуванні юридичних осіб не використовуються кредитні карти;
- за корпоративними кредитами істотно більше видів кредитування, ніж для роздрібного кредитування;
- рішення про видачу кредиту при кредитуванні корпоративного бізнесу приймає кредитний комітет банку (за роздрібними кредитами рішення приймає кредитний інспектор банку);

- моніторинг позичкової заборгованості за корпоративними кредитами проводиться частіше, ніж по роздрібних кредитах;
- такий вид забезпечення як банківська гарантія найчастіше видається суб'єкту корпоративного бізнесу.

Основними проблемами корпоративного кредитування, що чинять негативний вплив на розвиток процесів кредитування підприємств корпоративного бізнесу [13], є галузеві та валютні ризики, ризик незавершених інвестиційних проектів на тлі кризи, ризик затримки оплати контрагентами тощо.

Отже, кредитування підприємств вимагає багатосторонньої оцінки їх діяльності з метою отримання максимально об'єктивної оцінки ризиків.

1.2. Методи прийняття рішень на основі теорії нечітких множин

Елементи теорії нечітких множин можуть успішно застосовуватись для прийняття рішень в умовах невизначеності [l4].

Нехай U – повна множина, що охоплює всі об'єкти деякого класу. Нечітка підмножина F множини U, яка в подальшому називатиметься нечіткою множиною, визначається через функцію приналежності $\mu_F(u)$, $u \in U$. Ця функція відображає елементи U_i, множини U на множину дійсних чисел відрізка [0, 1], які вказують ступінь приналежності кожного елемента нечіткій множині F. Якщо повна множина U складається зі скінченного числа елементів u_i, $i = 1, 2, ..., n$, то нечітку множину F можна представити в наступному вигляді:

$$F = \mu_F(u_1)/u_1 + \mu_F(u_2)/u_2 + \cdots + \mu_F(u_n)/u_n, \qquad (1.1)$$

де «+» – об'єднання, а символ «/» показує, що значення μ_F відноситься до елементу u_i.

Над нечіткими множинами, як і над звичайними, можна виконувати математичні операції. Найважливішими із них є: доповнення множини (1.2), об'єднання (1.3) і перетин (1.4). Символ v позначає взяття максимуму, символ ∧ – взяття мінімуму.

$$\bar{F} = \sum_{i=1}^{n} \frac{(1-\mu_F(u_i))}{u_i}; \quad \mu_{\bar{F}} = 1 - \mu_F(u). \tag{1.2}$$

$$F \cup G = \sum_{i=1}^{n} \big(\mu_F(u_i) \vee \mu_G(u_i)\big)/u_i; \quad \mu_{F \cup G} = \mu_F(u) \vee \mu_G(u). \tag{1.3}$$

$$F \cap G = \sum_{i=1}^{n} \big(\mu_F(u_i) \wedge \mu_G(u_i)\big)/u_i; \quad \mu_{F \cap G} = \mu_F(u) \wedge \mu_G(u). \tag{1.4}$$

Нечітким відношенням R між повною множиною U і іншою повною множиною V називається підмножина прямого декартового добутку $U \times V$, який визначається наступним чином:

$$R = \sum_{i=1}^{l} \sum_{j=1}^{m} \mu_R(u_i, v_j)/(u_i, v_j), \tag{1.5}$$

де $U = \{u_1, u_2, ..., u_l\}$, $V = \{v_1, v_2, ..., v_m\}$.

Багатокритеріальний вибір альтернатив (БВА) на основі перетину нечітких множин.

Елементи теорії нечітких множин успішно застосо-вуються для. прийняття рішень. Експертні оцінки альтернативних варіантів за критеріями можуть бути представлені як нечіткі множини або числа, виражені за допомогою функцій приналежності. Для упорядкування нечітких чисел існує безліч методів, які відрізняються один від одного способом згортки і побудови нечітких відношень.

Нехай існує множина альтернатив $A = \{a_1, a_2, ..., a_m\}$ і безліч критеріїв $C = \{C_1, C_2, ..., C_n\}$, при цьому оцінки альтернатив по кожному i-му критерію представлені нечіткими множинами:

$$C_1 = \{\mu_{C_i}(a_1)/a_1, \mu_{C_i}(a_2)/a_2, \dots, \mu_{C_i}(a_m)/a_m\} \tag{1.6}$$

Правило вибору кращої альтернативи можна уявити як перетин нечітких множин, які відповідають критеріям:

$$D = C_1 \cap C_2 \cap \dots \cap C_n \tag{1.7}$$

Операція перетину нечітких множин може бути реалізована різними способами. Іноді перетин виконується як множення, але зазвичай цій операції відповідає взяття мінімуму:

$$\mu_D(a_j) = \min_{i=1,\dots,n} \mu_{C_i}(a_j),\ j = 1, \dots, m. \quad (1.8)$$

Кращою вважається альтернатива a^*, що має найбільше значення функції приналежності:

$$\mu_D(a^*) = \max_{j=1,\dots,m} \mu_D(a_j). \quad (1.9)$$

Якщо критерії C_i мають різну важливість, то їх внесок в загальне рішення можна уявити як зважений перетин:

$$D = C_1^{a_1} \cap C_2^{a_2} \cap \dots \cap C_n^{a_n}, \quad (1.10)$$

де a_i – вагові коефіцієнти відповідних критеріїв, які повинні відповідати таким вимогам:

$$a_i \geq 0; i = 1, \dots, n;\ (1/n) \sum_{i=1}^{n} a_i = 1. \quad (1.11)$$

Коефіцієнти відносної важливості можна визначити, використовуючи процедуру попарного порівняння критеріїв.

БВА на основі нечіткого відношення переваги.

Даний метод прийняття рішень передбачає побудову множини недомінуючих альтернатив на основі нечіткого відношення переваги [16].

Нехай задано множину альтернатив A і кожна альтернатива характеризується кількома критеріями якості з номерами $j == i,\dots, m$. Інформація про попарне порівняння альтернатив за кожним критерієм якості j представлена у формі відношення переваги R_j. Таким чином є m відношень переваги R_j на множині A. Потрібно вибрати кращу альтернативу з множини $\{A, R_1, \dots, R_m\}$.

Визначення 1. Нечітким відношенням R на множині A називається нечітка підмножина декартового добутку $A \times A$, що характеризує функцію приналежності μ_R: $A \times A \rightarrow [0, 1]$. Значення μ_R (a, b) цієї функції розуміється як ступінь виконання відношення $a \wedge b$.

Визначення 2. Нечітким відношенням переваги на A називається будь-яке задане на цій множині рефлексивне нечітке відношення, функція приналежності якого обчислюється таким чином:

$$\mu_R s(a,b) = \begin{cases} \mu_R(a,b) - \mu_R(b,a),\ \mu_R(a,b) \geq \mu_R(b,a); \\ 0,\ \mu_R(a,b) < \mu_R(b,a). \end{cases} \quad (1.12)$$

Визначення 3. Нехай A – множина альтернатив і μ_R – задане на ній нечітке відношення переваги. Нечітка підмножина недомінуючих альтернатив множини (A, μ_R) описується функцією приналежності

$$\mu_R^{\text{НД}} = 1 - \sup_{a,b\epsilon A}\left(\mu_R s(b,a) - \mu_R s(a,b)\right), a\epsilon A. \quad (1.13)$$

Визначення 4. Чітко недомінуючими називаються альтернативи, для яких $\mu_R^{\text{НД}}(a) = 1$, а множина таких альтернатив

$$A^{\text{ЧНД}} = \{a | a\epsilon A,\ \mu_{R^{\text{НД}}}(a) = 1\}. \quad (1.14)$$

Визначення 5. Носієм нечіткої множини B з функцією приналежності $\mu_B(a)$ є множина $\{a|a \in A,\ \mu_B > 0\}$. Процедура вирішення завдання вибору виконується в кілька кроків.

1. Будується нечітке відношення Q_1 яке є перетином вихідних відношень переваги:

$$\mu_{Q_1}(a,b) = min\left(\mu_{R_1}(a,b), \dots, \mu_{R_m}(a,b)\right). \quad (1.15)$$

і визначається нечітка підмножина недетермінованих альтернатив в множині (A, μ_{Q1}):

$$\mu_{Q_1}^{\text{НД}}(a) = 1 - \sup_{b\epsilon A}\left(\mu_{Q_1}(b,a), \dots, \mu_{Q_1}(a,b)\right). \quad (1.16)$$

2. Будується нечітке відношення Q_2 і визначається нечітка підмножина недетмінованих альтернатив в множині (A, μ_{Q2}) :

$$\mu_{Q_2}(a,b) = \sum_{j=1}^{m} w_j \mu_{R_j}(a,b). \quad (1.17)$$

$$\mu_{Q_2}^{\text{НД}}(a) = 1 - \sup_{b\epsilon A}\left(\mu_{Q_2}(b,a) - \mu_{Q_2}(a,b)\right). \quad (1.18)$$

Ця функція впорядковує альтернативи за ступенем їх недермінованості. Числа w_j в наведеній вище згортці представляють собою коефіцієнти відносної важливості критеріїв, що розглядаються, для яких виконуються наступні умови:

$$\sum_{j=1}^{m} w_j = 1,\ w_j \geq 0,\ j = \overline{1,m} \quad (1.19)$$

3. Відшукується перетин множин $\mu_{Q_1}^{\text{НД}}$ і $\mu_{Q_2}^{\text{НД}}$:

$$\mu^{\text{НД}}(a) = min\left(\mu_{Q_1}^{\text{НД}}(a), \mu_{Q_2}^{\text{НД}}(a)\right). \quad (1.20)$$

Раціональним вважається вибір альтернатив з множини:

$$A^{\text{НД}} = \{a' | a' \epsilon A,\ \mu^{\text{НД}}(a') = \min_{a \epsilon A} \mu^{\text{НД}}(a)\}. \quad (1.21)$$

Найбільш раціональною альтернативою з множини $A^{НД}$ є та, яка має максимальну ступінь недомінованості.

БВА з використанням правила нечіткого виведення.

Даний метод багатокритеріального вибору альтернатив на основі композиційного правила агрегування описів альтернатив з інформацією про вподобання особи, що приймає рішення, які задані у вигляді нечітких суджень [21].

Нехай U – безліч елементів, A – його нечітка підмножина, ступінь приналежності елементів до якого є число з одиничного інтервалу [0, 1]. Підмножини A_j є значеннями лінгвістичної змінної X.

Припустимо, що множина рішень характеризується набором критеріїв $x_1, x_2, ..., x_p$, тобто лінгвістичних змінних, заданих на базових множинах $u_1, u_2, ... u_p$ відповідно. Набір з декількох критеріїв з відповідними значеннями характеризує вподобання особи, що приймає рішення. Змінна S («задовільність») також є лінгвістичною.

Нехай є висловлювання d_1:

«Якщо $x_1 = A_{1i}, x_2 = A_{2i}, ... x_p = A_{pi}$, то $S = B_i$» (1.22)

Операції перетину нечітких множин відповідає знаходженню мінімуму їх функцій приналежності:

$$\mu_{A_i}(v) = \min_{v \epsilon V} \left(\mu_{A_{i1}}(u_1), \mu_{A_{i2}}(u_2), \dots, \mu_{A_{ip}}(u_p) \right). \quad (1.23)$$

Тут $V = U_1 \times U_2 \times ... \times U_p$; $v = (u_1, u_2 ..., u_p)$; $\mu_{A_{ij}}(u_j)$ – значення приналежності елемента до нечіткої множини A_{ij}. Тоді вираз (1.21) можна записати у вигляді:

$$d_i: \text{"Якщо } x = A_i \text{ то } S = B_i\text{"}. \quad (1.24)$$

Для надання загальності судженням позначимо базові множини U і V через W. Тоді A_i – нечітка підмножина W, в той час як B_i – нечітка підмножина одиничного інтервалу I. Для представлення правил

використовується операція імплікації, для якої запропоновано різні способи нечіткої реалізаціі [31]. Нечітка імплікація Лукасевича має вигляд:

$$\mu_H(w,i) = \min_{w\epsilon W}\left(1,\left(1-\mu_A(w)+\mu_B(i)\right)\right), \quad (1.25)$$

де H – нечітка підмножина на $W \times I$, $w \in W$, $i \in I$.

Аналогічним чином вирази d_1, d_2,..., d_q перетворюються в множини H_1, H_2, ..., H_q. Їх перетином є множина D:

$$D = H_1 \cap H_2 \cap ... \cap H_q \quad (1.26)$$

і для кожного $(w, i) \in W \times I$

$$\mu_D(w,i) = \min_{w\epsilon W}\left(\mu_{H_i}(w,i)\right),\ j = \overline{1,q}. \quad (1.27)$$

Задовільність альтернативи, яка описується нечіткою підмножиною A з W, визначається на основі композиційного правила виведення:

$$G = A \circ D, \quad (1.28)$$

де G – нечітка підмножина інтервалу I.

Тоді

$$\mu_G(i) = \max_{w\epsilon W}(\min \mu_A(w)\mu_D(w,i)). \quad (1.29)$$

Зіставлення альтернатив відбувається на основі точкових оцінок. Для нечіткої множини $C \subset I$ визначаємо α-рівневу множину ($\alpha \in [0, 1]$):

$$C_\alpha = \{i \mid \mu_c(i) \geq \alpha \in I\}. \quad (1.30)$$

Для кожного C_α можна обчислити середнє число елементів – $M(C_\alpha)$:

- для множини з n елементів:

$$M(C_\alpha) = \sum_{j=1}^{n} i_j \,|n;\ i_j \epsilon C_\alpha; \quad (1.31)$$

- для $C_\alpha = \{a \leq i \leq b\}$:

$$M(C_\alpha) = \frac{a+b}{2}; \quad (1.32)$$

- для $C_\alpha = \prod_{j=1}^{n}\{a_j \leq i \leq b_j\}$:

$$M(C_\alpha) = \frac{\sum_{j=1}^{n}\frac{a_j+b_j}{2}(b_j-a_j)}{\sum_{j=1}^{n}(b_j-a_j)} \quad (1.33)$$

при $0 \leq a_1 \leq b_1 \leq a_2 \leq b_2 \leq ... \leq a_n \leq b_n \leq 1$.

Тоді точкове значення для множини C можна записати у вигляді:

$$F(C) = \frac{1}{\alpha_{max}} \int_0^{\alpha_{max}} M(C_\alpha) d\alpha, \quad (1.34)$$

де α_{max} – максимальне значення в множині C.

При виборі альтернатив для кожної з них знаходиться задоволеність і обчислюється відповідна точкова оцінка. Кращою вважається альтернатива з найбільшим її значенням.

БВА на основі адитивної згортки.

В даному методі [17] експертні переваги представлені за допомогою нечітких чисел, що мають функції приналежності трикутного виду (рис. 1.3).

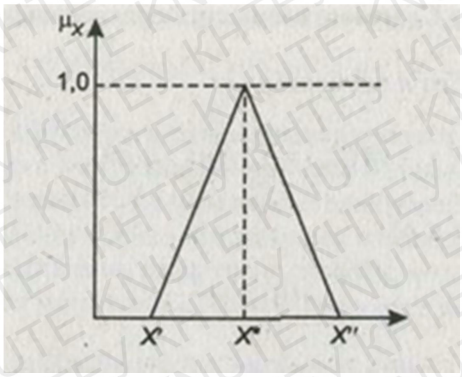

Рис. 1.3. Трикутне представлення нечіткого числа

Нехай є множина альтернатив $A = \{a_1, a_2, ..., a_m\}$ і множина критеріїв C $= \{c_1, c_2, ..., c_n\}$, при цьому оцінка j-ї альтернативи по i-му критерію представлена нечітким числом R_{ij}, а відносна важливість i-го критерію задається коефіцієнтом α i = 1,2 ...,n. Якщо коефіцієнти a нормовані, то зважена оцінка j-ї альтернативи обчислюється за формулою

$$R_j = \sum_{i=1}^{n} \alpha_i R_{ij}. \quad (1.35)$$

Якщо функції приналежності $\mu_{Rij}(r_{ij})$ і $\mu_{\alpha i}(\alpha_i)$ мають трикутний вигляд, то для них, як і для нечіткого числа X, вершина X*, а також ліва X' і права X'' межі визначаються наступними співвідношеннями:

$$\begin{gathered} \forall\delta: \mu(X') = 0;\ \mu(X' - \delta) = 0;\ \mu(X' + \delta) \neq 0; \\ \forall\delta: \mu(X'') = 0;\ \mu(X'' - \delta) \neq 0;\ \mu(X' + \delta) = 0;\ \mu(X^*) = 1. \end{gathered} \quad (1.36)$$

Зважена оцінка j-ї альтернативи R_j є результатом лінійної комбінації нечітких чисел і також матиме функцію приналежності трикутного виду. Вершину і межі нечіткого числа Z == $X \times Y$, отриманого в результаті операцій

додавання або множення (символ $\times$ позначає узагальнену операцію), можна обчислити таким чином:

$$Z'=X' \times Y';\ Z''=X'' \times Y'';\ Z^*=X^* \times У. \quad (1.37)$$

Ранжування альтернатив з використанням отриманих зважених оцінок можливе на основі їх нечіткої композиції:

$$\mu_J(j) = \sup_{r_1,r_2,\ldots,r_m;\, r_k \geq r_j} \min_{j=1,\ldots,m} \mu_{R_j}(r_j). \quad (1.38)$$

Тут $\mu_J(j)$ – нечітка множина альтернатив, що відповідають поняттю «краща альтернатива». Кращою вважається альтернатива, що має найбільше значення $\mu_J(j)$.

Пріоритет кожної альтернативи обчислюється шляхом вибору мінімуму серед точок перетину правої межі відповідного їй нечіткого числа R_j з межами нечітких чисел, що представляють зважені оцінки альтернатив, розташованих правіше на числовій осі (що задовільняють умові $r_k > r_j$). При цьому мається на увазі, що права межа області визначення нечітких чисел відповідає найкращим оцінками, а ліва – найгіршим.

Ранжування альтернатив на множині лінгвістичних векторних оцінок.

Задано множину альтернатив $A == \{a_1, a_2, \ldots, a_m\}$ і множину відповідних результатів $S = \{s_1, s_2, \ldots, s_m\}$. Кожен результат s_j характеризується альтернативою a_i і вектором лінгвістичних оцінок на множині критеріїв $K = \{K_1, K_2, \ldots. K_n\}$. Безліч лінгвістичних векторних оцінок результатів $K = \{K(s_1), K(s_2), \ldots, K(s_m)\}$ можна впорядкувати, ввівши функцію приналежності нечіткого відношення порядку $\mu_{\geq}: K \times K \rightarrow [0, 1]$. Для i-го критерію позначимо $\mu^i_{\geq}(K_i(s_j), K_i(s_k))$ через $\mu^i_{\geq}(s_j, s_k)$. Значення цієї функції можна обчислити за формулою:

$$\mu^i_{\geq}(s_j, s_k) = 1 - \mu^i_{<}(s_j, s_k) = \mu^i_{>}(s_j, s_k) + \mu^i_{=}(s_j, s_k). \quad (1.39)$$

Ступінь істинності $\mu < (s_j, s_k)$ нечіткого виразу $s_j < s_k$ можна визначити як ймовірність того, що точне значення s_j буде набагато меншим від точного

значення s_k. Припускаючи, що результати є незалежними випадковими величинами, відношення $\mu_<(s_j, s_k)$ можна представити у вигляді:

$$\mu^i_<(s_j, s_k) = \sum_{i=1}^{n-1}\left(v_{sj}(x_i)(1 - w_{sk}(x_{i+1}))\right) \quad (1.40)$$

де $v_s(x)$ – ймовірність того, що в якості точного значення нечіткого числа s використовується величина x; $w_s(x)$ – ймовірність того, що в якості точного значення s використовується величина $y < x$:

$$v_S(x) = \mu_S(x) / \sum_{y \epsilon S} \mu_S(y); \;\; w_S(x) = \sum_{y \epsilon S, y<x} v_S(y). \quad (1.41)$$

Векторні оцінки можуть бути впорядковані на основі функції приналежності:

$$\mu_\geq \left(K(s_j), K(s_k)\right) = \times_{i \epsilon n} \mu^i_\geq(s_j, s_k), \quad (1.42)$$

де х – позначає символ узагальненої операції.

Так як між безліччю альтернатив та результатів існує взаємно однозначна відповідність, функцію приналежності нечіткого відношення переваги на множині альтернатив можна представити у вигляді:

$$\mu^F_\geq(a_j, a_k) = \mu_\geq \left(K(s_j), K(s_k)\right). \quad (1.43)$$

Розв'язок задачі з використанням даного методу включає наступні основні кроки:

- обчислення функцій приналежності $\mu_<$ з використанням співвідношень (1.40);
- побудова нечіткого відношення порядку $\mu_\geq$;
- мінімізація відношення $\mu_\geq$;
- визначення відношень переваги на множині альтернатив і виявлення кращої альтернативи. Для цього обчислюється відношення переваги між альтернативою a_j і всіма іншими альтернативами, функція приналежності якого має вигляд:

$$\mu^F_\geq(a_j; \{a_k\}, k \epsilon I_j) = \times_{k \epsilon \; j} \mu_\geq jk, \;\; j \epsilon m, \quad (1.44)$$

де I_j – множина індексів альтернатив, з якими може порівнюватись j-а альтернатива.

Розв'язання задачі ранжування можна описати співвідношеннями:

$$r_j < r_l \Leftrightarrow \mu_{\geq}^{F}(a_j;\{a_k\},k\epsilon I_j) > \mu_{\geq}^{F}(a_l;\{a_k\},k\epsilon I_l);$$
$$r_j = r_l \Leftrightarrow \mu_{\geq}^{F}(a_j;\{a_k\},k\epsilon I_j) = \mu_{\geq}^{F}(a_l;\{a_k\},k\epsilon I_l), \quad (1.45)$$

де r_j – ранг альтернативи.

Найкраща альтернатива має найнижчий ранг [4].

1.3. Характеристики систем управління банківською діяльністю та місце задач кредитування у їх складі

Автоматизована банківська система (далі АБС) – це система, яка функціонує на основі ЕОМ та інших технічних засобів, що забезпечують процеси збору, реєстрації, передачі, обробки, збереження та актуалізації даних для розв'язання завдань управління банківською діяльністю.

Основними функціональними підсистеми АБС:

- операційний день банку (ОДБ);
- управління валютними операціями (Валютні операції);
- управління депозитами (Депозити);
- управління цінними паперами (Цінні папери);
- управління касою (Каса);
- внутрібанківський облік (Внутрішній облік);
- управління звітністю (Звітність банку);
- управління розрахунками з використанням пластикових карток (Карткові операції);
- звітність, аналіз діяльності банку (Аналіз);
- управління кредитними ресурсами (Кредити).

ОДБ – це базова підсистема, що має обов'язково функціонувати в кожному комерційному банку.

У комерційних банках, що мають дозвіл на виконання операцій з іноземною валютою, повинна функціонувати підсистема «Валютні операції». В межах цієї підсистеми обов'язково повинен функціонувати комплекс задач «Валютний операційний день», що забезпечує введення та обробку валютних платіжних документів, відкриття та закриття валютних рахунків, конвертацію валют, ведення рахунків покриття, формування балансу та інші

операції з іноземною валютою. Операції з валютою можуть бути автоматизованим комплексним мультивалютним ОДБ, який здатний працювати як з національною, так і з будь-якою іншою валютою.

Основними завданнями підсистеми «Депозити» є облік операцій з готівкою, облік безготівкових операцій, облік цінних бланків, нарахування відсотків за депозитними рахунками, а також формування звітних форм щодо роботи з депозитними вкладами. В деяких АБС вона не виділяється в окрему функціональну підсистему, а інтегрується в комплекс, який має назву «Управління кредитно-депозитними операціями».

У підсистемі «Цінні папери» здійснюється автоматизація обліку операцій з власними акціями банку, з державними цінними паперами, з іншими цінними паперами (акціями підприємств, векселями, сертифікатами), автоматизація управління портфелем цінних паперів, моделювання та прогнозування стану фондового ринку.

Підсистема «Каса» необхідна для обліку готівки та організації роботи обмінних пунктів.

Підсистема «Внутрішній облік» включає до свого складу задачі, пов’язані з обліком у самому банку. До цієї підсистеми належать такі основні задачі: облік праці і нарахування заробітної плати працівникам банку; облік власних основних засобів банку; облік нематеріальних активів; облік амортизації основних засобів та нематеріальних активів; облік господарських і експлуатаційних витрат.

У підсистемі «Звітність банку» формується бухгалтерська, фінансова та статистична звітність про діяльність комерційного банку.

Метою роботи підсистеми «Карткові операції» є автоматизація безготівкових розрахунків з фізичними особами.

Підсистема «Аналіз» акумулює у своєму складі аналітичні задачі, які належать до класу OLAP.

В межах підсистеми «Кредити» працівники кредитного відділу мають можливість виконувати аналіз фінансового стану позичальника, визначення

його кредитоспроможності та оцінку ризику при кредитуванні, формування та облік кредитних договорів, ведення та коригування розпоряджень на оплату кредитів, ведення та коригування строкових зобов'язань на погашення кредиту, ведення та коригування відсоткових ставок і графіків оплати відсотків по кредитному договору, нарахування відсотків по кредиту та облік їх сплати, облік та контроль погашення кредитної заборгованості і аналіз кредитного портфелю. Ця підсистема є інтегрованою з іншими функціональними підсистемами банку, зокрема з ОДБ, у якій виконують бухгалтерські проведення при наданні кредиту та при погашенні суми основного боргу і відсотків по ньому [3].

Отже, управління процесами, що є об'єктом дослідження випускного кваліфікаційного проекту, виконуються у підсистемі «Кредити».

РОЗДІЛ 2.
ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК)

2.1. Постановка задачі

З розвитком ринкових відносин процес кредитування банками підприємств пов'язаний з численними факторами ризику, здатними спричинити за собою непогашення позики у встановлений термін. При аналізі кредитоспроможності позичальника визначається можливість своєчасного і повного погашення заборгованості за позикою; ступінь ризику, яку банк готовий взяти на себе; розмір кредиту, який може бути наданий в конкретній ситуації; умови надання кредиту.

В сучасних умовах аналіз кредитоспроможності пов'язаний не тільки з оцінкою платоспроможності клієнта на певну дату, а й з виявленням найбільш бажаних позичальників, прогнозуванням їх фінансової стійкості в перспективі, урахуванням можливих ризиків за кредитними операціями. Проведення такого всебічного аналізу дозволяє банку більш ефективно управляти кредитними ресурсами і отримувати прибуток.

Застосовувані банками методи в області кредитування базуються на даних бухгалтерських звітів, тому вони дозволяють лише оцінити кредитоспроможність позичальника, не забезпечуючи вибору найбільш оптимального позичальника з метою мінімізації факторів ризику для банку і найбільш ефективного планування своєї діяльності в майбутньому.

2.2. Метод максимінної згортки (з урахуванням однакової ваги критеріїв) у сфері банківського кредитування

Метод передбачає виконання трьох етапів на основі даних, спосіб розрахунку яких предсталено у підрозділі 2.3.

Етап 1. Побудова функцій приналежності [14, 31], що відповідають поняттям «кращий коефіцієнт абсолютної ліквідності», «бажаний проміжний коефіцієнт покриття», «найкращий коефіцієнт рентабельності» і т.д. (рис.

2.1). Побудову таких функцій проводять експерти, які мають знання в області кредитування підприємств різного функціонального призначення.

Етап 2. Визначаються конкретні значення функції приналежності за критеріями якості $F_1, \ldots, F_5$. На рисунку 2.1 показано значення функцій приналежності, що відповідають альтернативам, які розглядаються. Нечіткі множини для п'яти критеріїв, що розглядаються і включають чотири аналізовані альтернативи, мають такий вигляд:

$$\begin{aligned} \mu_{F_1} &= 0{,}61/0{,}154 + 0{,}41/0{,}102 + 0{,}33/0{,}084 + 0{,}46/0{,}140; \\ \mu_{F_2} &= 1{,}0/1{,}297 + 0{,}71/0{,}71 + 0{,}59/0{,}59 + 0{,}57/0{,}57; \\ \mu_{F_3} &= 1{,}0/2{,}78 + 0{,}91/2{,}27 + 0{,}75/1{,}86 + 0{,}51/1{,}27; \\ \mu_{F_4} &= 1{,}0/0{,}75 + 0{,}96/0{,}72 + 0{,}94/0{,}71 + 0{,}90/0{,}68; \\ \mu_{F_5} &= 0{,}93/0{,}28 + 0{,}38/0{,}115 + 0{,}5/0{,}15 + 0{,}4/0{,}12. \end{aligned} \qquad (2.1)$$

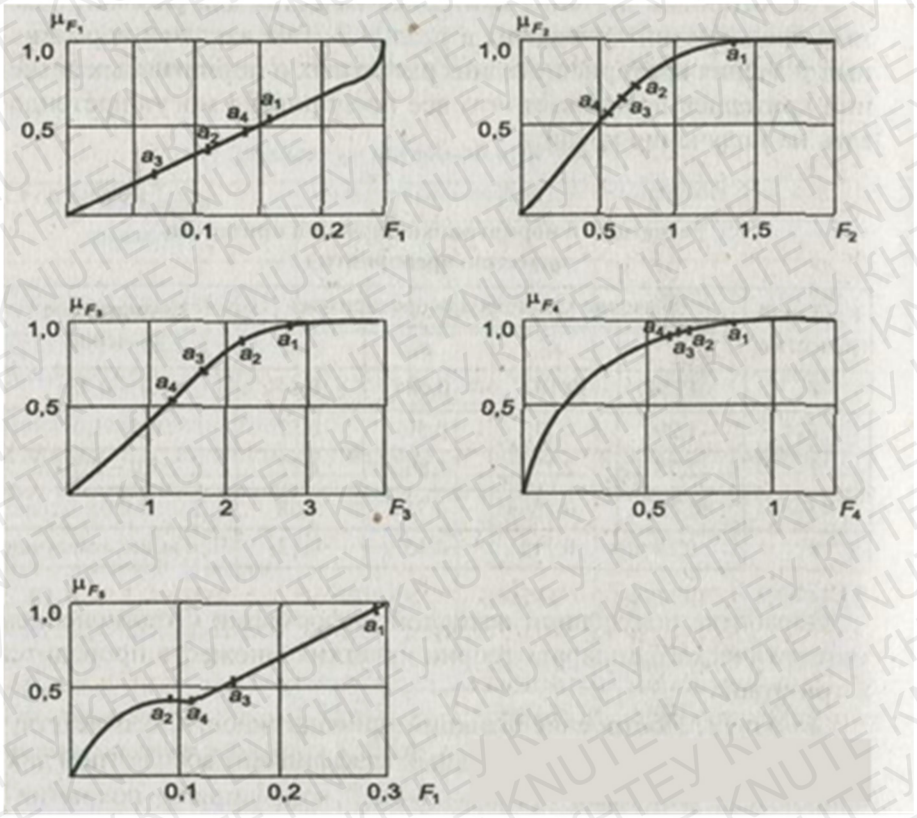

Рис. 2.1. Функції приналежності критеріїв якості

Етап 3. Проводиться згортка наявної інформації з метою виявлення кращої альтернативи. Множина оптимальних альтернатив B визначається шляхом перетину нечітких множин, що містять оцінки альтернатив за критеріями вибору.

Якщо критерії, за якими здійснюється вибір варіантів, мають однакову важливість для особи, що приймає рішення, то правило вибору кращого варіанту має вигляд:

$$B = F_1 \cap F_2 \cap F_3 \cap F_4 \cap F_5. \quad (2.2)$$

Оптимальною вважається альтернатива з максимальним значенням функції приналежності до множини B. Операція перетину нечітких множин відповідає вибору мінімального значення для j-ї альтернативи:

$$\mu_B(a_j) = min_{F_{i_i}}(a_j) \quad (2.3)$$

Для даної задачі множина оптимальних альтернатив буде формуватися наступним чином:

$$B = \begin{Bmatrix} \min\{0{,}61;\ 1{,}0;\ 1{,}0;\ 1{,}0;\ 0{,}93\}, \\ \min\{0{,}41;\ 0{,}71;\ 0{,}91;\ 0{,}96;\ 0{,}38\} \\ \min\{0{,}33;\ 0{,}59;\ 0{,}75;\ 0{,}94;\ 0{,}50\} \\ \min\{0{,}46;\ 0{,}57;\ 0{,}51;\ 0{,}90;\ 0{,}40\} \end{Bmatrix}. \quad (2.4)$$

Результуючий вектор пріоритетів альтернатив має наступний вигляд:

$$\frac{\max \mu_B(a_j)}{j} = \max\{0{,}61;\ 0{,}38;\ 0{,}33;\ 0{,}4\}. \quad (2.5)$$

Таким чином, найкращою альтернативою є a_1, якій відповідає значення 0,61. На другому, третьому і четвертому місцях знаходяться відповідно $a_4 \rightarrow 0{,}4$, $a_2 \rightarrow 0{,}38$, $a_3 \rightarrow 0{,}33$.

2.3. Аналітичний апарат розрахунку коефіцієнтів кредитоспроможності позичальників

Апарат розрахунку коефіцієнтів буде розглянуто на конкретному прикладі.

У цій задачі підприємства є альтернативами, з яких належить зробити вибір кращої. Альтернативи позначаються через $a_1, ..., a_4$.

Для оцінки кредитоспроможності підприємств-позичальників використовуються дані їх бухгалтерської звітності (табл. 2.1).

Таблиця 2.1. Дані бухгалтерської звітності

Фінансовий показник	Значення показника для підприємства, тис. грн.			
	a_1	a_2	a_3	a_4
Грошові засоби (ГС)	229,1	946,2	947,0	1442,9
Короткострокові фінансові вкладення (КФВ)	394,1	462,7	466,4	2066,0
Дебіторська заборгованість (ДЗ)	4639,8	8391,4	8514,5	10908,2
Запаси і витрати (ЗВ)	6028,1	21557,6	21370,4	17424,5
Власний капітал (ВК)	12395,8	35247,8	41244,2	53939,4
Короткострокові зобов'язання (ЗКс)	4058,1	13834,9	16827,1	25028,3
Підсумок балансу (ПБ)	16453,9	49082,7	58071,3	78967,7
Валова виручка (ВВ)	59438,9	38567,9	43589,5	28343,6
Прибуток (П)	16642,9	4442,5	65384,2	3401,2

На підставі цих даних розраховуються фінансові коефіцієнти, що характеризують кредитоспроможність позичальників:

- коефіцієнт абсолютної ліквідності (F_1),
- проміжний коефіцієнт покриття (F_2),
- загальний коефіцієнт покриття (F_3),
- коефіцієнт фінансової незалежності (F_4)
- коефіцієнт рентабельності продукції (F_5).

Перераховані коефіцієнти є критеріями якості кредитоспроможності підприємств і розраховуються за такими формулами:

$$F_1 = \frac{\text{ГС+КФВ}}{\text{ЗКс}};\ F_2 = \frac{\text{ГС+КФВ+ДЗ}}{\text{ЗКс}};\ F_3 = \frac{\text{ГС+КФВ+ДЗ+ЗВ}}{\text{ЗКс}};$$
$$F_4 = \frac{\text{ВК}}{\text{ПБ}};\ F_5 = \frac{\text{П}}{\text{ВВ}}. \quad (2.6)$$

Розраховані значення критеріїв якості для підприємств, що розглядаються, наведено в таблиці 2.2. У ній також представлено нормативні значення критеріїв. Аналіз розрахункових та нормативних значень критеріїв показує, що всі підприємства можуть претендувати на отримання кредиту.

Таблиця 2.2. Розрахункові і нормативні значення критеріїв якості підприємств

Критерій якості	Значення критерію для підприємства				Нормативні значення
	a_1	a_2	a_3	a_4	
F_1	0.154	0.102	0.084	0.140	0.1-0.25
F_2	1.297	0.71	0.59	0.57	0.5-1.0
F_3	2.78	2.27	1.86	1.27	1.0-2.5
F_4	0.75	0.72	0.71	0.68	0.6
F_5	0.28	0.115	0.15	0.12	Чим вище, тим краще

На основі розрахованих значень критеріїв і математичного апарату теорії нечітких множин здійснюються наступні етами багатокритеріального вибору методом максімнної згортки.

2.4. Методи побудови функції приналежності. Лінійна інтерполяція

При побудові функцій належності для нечітких множин слід дотримуватися деяких правил, які визначаються характером невизначеності, що має місце при побудові конкретних нечітких моделей.

З практичної точки зору з кожною нечіткою множиною зручно асоціювати деяку властивість, ознаку або атрибут, які характеризують розглянуту сукупність об’єктів універсуму. При цьому за аналогією з класичними множинами аналізована властивість може породжувати деякий предикат, який цілком природно назвати нечітким предикатом. Даний нечіткий предикат може приймати не одне з двох значень істинності, а цілий континуум значень істинності, які для зручності вибираються з інтервалу [0, 1]. При цьому значенню «істина» відповідає число 1, а значенню «брехня» – число 0. Чим більшою мірою елемент $x \in X$ володіє даною властивістю, тим ближче до 1 повинно бути значення істинності відповідного нечітко предиката.

В загальному випадку задання нечіткої множини із залученням спеціальної властивості еквівалентне заданню такої функції приналежності,

яка змістовно представляє ступінь істинності відповідного одномісного нечітко предикату. Найбільшого поширення при побудові функцій приналежності нечітких множин отримали прямі і непрямі методи.

Прямі методи побудови функцій приналежності.

У прямих методах експерт або група експертів просто задають для кожного $x \in X$ значення функції приналежності $\mu_{\widetilde{A}}(x)$. Як правило, прямі методи побудови функцій приналежності використовуються для таких властивостей, які можуть бути виміряні якоюсь кількісною шкалою. Наприклад, такі фізичні величини, як швидкість, час, відстань, тиск, температура та інші мають відповідні одиниці і еталони для свого виміру. При цьому доцільно обмежити розгляд тільки тими значеннями величин, які мають фізичний зміст в контексті розв'язуваної задачі.

При прямій побудові функцій приналежності слід враховувати ту обставину, що теорія нечітких множин не вимагає абсолютно точного задання функцій приналежності. Найчастіше буває досить зафіксувати лише найхарактерніші значення і вид (тип) функції приналежності.

Процес побудови або задання нечіткої множини на основі деякого відомого заздалегідь кількісного значення вимірюваної ознаки навіть отримав спеціальну назву – фазифікація або приведення до нечіткості.

Непрямі методи побудови функцій приналежності.

Як правило, непрямі методи визначення значень функції приналежності використовуються в тих випадках, коли відсутні очевидні вимірні властивості, які можуть бути використані для побудови нечітких моделей розглянутої предметної області.

Серед непрямих методів найбільш відомий так званий метод попарних порівнянь. Цей метод використовується для скінченних нечітких множин і заснований на наступному припущенні: якби значення шуканої функції приналежності були відомі і дорівнювали значенням $\mu_{\widetilde{A}}(x)$ для $i \in \{1, 2, ..., n\}$, то попарні порівняння відповідних елементів носія нечіткої множини можна було б представити у вигляді матриці $\widetilde{A}$ з елементами a_{ij}, при цьому елементи

цієї матриці дорівнюють $a_{ij} = \mu_{\tilde{A}}(x_i)/\mu_{\tilde{A}}(x_j)$, де символ "/" позначає операцію ділення.

На практиці буває простіше спочатку побудувати матрицю А в припущенні, що її діагональні елементи повинні бути рівні 1, а симетричні відносно головної діагоналі елементи повинні бути взаємно зворотними, тобто $a_{ij} = 1/a_{ji}$. Остання умова означає, що якщо ступінь приналежності одного з елементів оцінюється в a разів сильніше ступеня приналежності іншого, то ступінь приналежності другого елемента повинна бути в *1/a* раз сильнішою ступеня приналежності першого елемента.

У цьому випадку задання побудови функції приналежності зводиться до знаходження такого вектора w, який є розв'язком рівняння $A \cdot w = \lambda_{max} w$, де λ_{max} – найбільше власне значення матриці A. Оскільки всі значення елементів матриці A позитивні згідно побудови, то розв'язок даного рівняння існує і є позитивним.

Власне процес попарного порівняння елементів може бути заснований на суб'єктивній інтуїції або на виконанні певної послідовності алгоритмічних чи логічних дій. При цьому окремі елементи універсуму можуть використовуватись в якості еталонів або ж всі елементи можуть бути розділені на групи з подальшим порівнянням цих груп між собою.

З алгоритмічних процедур найбільшу популярність здобули методи ітеративного уточнення значень функцій приналежності, засновані на нейронних мережах і генетичних алгоритмах. Логічні процедури використовують методи індуктивного навчання і побудови нечітких метаправил. Іноді застосовуються методи обробки статистичних даних, факторного і дискримінантного аналізу з метою виділення значимих ознак для подальшого порівняння елементів розглянутого універсуму.

На закінчення слід зазначити, що в разі нестачі інформації про особливості функцій приналежності нечітких змінних рекомендується починати побудову нечіткої моделі з використання найбільш простих форм

функції приналежності, а саме кусково-лінійних функцій. Надалі їх характер може бути уточнений і врахований на етапі корекції нечіткої моделі [18].

Лінійна інтерполяція.

Інтерполяція – це метод знаходження невідомих проміжних значень деякої функції за наявним дискретним набором її відомих значень. Типовим прикладом такої функції є часовий ряд, значення якого – це спостереження, зафіксовані через певний інтервал часу. Наприклад, якщо спостереження за ходом досліджуваного бізнес-процесу (скажімо, продажів) реєструвалися в останній день кожної декади, то при необхідності оцінити значення всередині даного інтервалу потрібно виконати інтерполяцію (рис. 2.2).

Точки $x_1, x_2, \ldots, x_n$ називаються вузлами інтерполяції, їх сукупність – інтерполяційно. сіткою, а відстань між її сусідніми вузлами – кроком інтерполяції, який може бути як рівномірним, так і нерівномірним. Завдання полягає в пошуку інтерполюючої функції $F(x_i) = y_i$

Іншими словами, інтерполяція дозволяє дізнатися які значення приймає функція в точках, що не є її вузлами.

В даний час існує безліч різних методів інтерполяції. Вибір найбільш гідного із них визначається вимогою до точності, обчислювальної складності, гладкості інтерполюючої функції, кількості точок даних і т.д.

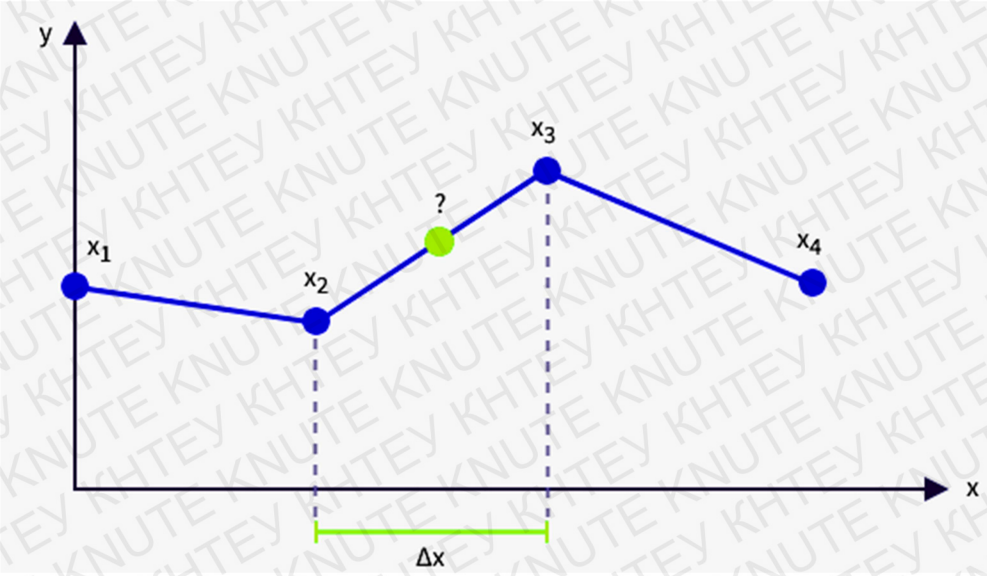

Рис. 2.2. Пошук невідомих значень на основі набору наявних дискретних значень методом інтерполяції

Найбільш простим методом є лінійна інтерполяція, коли передбачається, що проміжні точки лежать на прямих, що з'єднують її вузли (як показано на рис. 2.2). Інтерполююча функція в цьому випадку має вигляд:

$$f(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i). \quad (2.7)$$

Очевидно, що якщо спостереження фіксувалися рідко і крок інтерполяції великий, то даний метод може виявитися занадто грубим. Тому часто використовують інтерполяцію поліномами (формула Ньютона, поліноми Лагранжа), сплайн-функціями і т.д.

У технологіях аналізу даних інтерполяція використовується для відновлення пропущених значень, а також заміни аномальних.

Потрібно зазначити, що під час проведення аналітичних розрахунків було використано прямі методи побудови функцій приналежності. Це пов'язано із тим, що експерт безпосередньо задає правила визначення значень функції приналежності, які характеризують критерій оцінки банку.

РОЗДІЛ 3.
РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ВІДБОРУ ПІДПРИЄМСТВ ДЛЯ КРЕДИТУВАННЯ (КСВПК)

3.1. Загальна характеристика КСВПК

Технології, на базі яких побудовано комп'ютерну систему

Сайт написано на базі мови програмування JavaScript. На ній побудовані фреймворки та бібліотеки, які було використано у розробці проекту, і вона дозволяє створювати як серверну частину сайту, так і клієнтську.

Набір усіх використаних у розробці технологій та засобів відображено на рисунку 3.1.

Рис. 3.1. Середовище розробки та стек технологій

Основою проекту є стек технологій MERN (MongoDB, Express.js, React, Node.js). Із них серверна частина побудована на:

- MongoDB – документоорієнтована система управління базами даних;
- Express.js – бібліотека, яка має у своєму розпорядженні безліч службових HTTP-методів і проміжних оброблювачів, що дозволяють створити надійний API;

- Node.js – програмна платформа, побудована на рушії V8 (здійснює трансляцію JavaScript в машинний код), і виконує роль веб-сервера та забезпечує підключення зовнішніх бібліотек, написаних на різних мовах [26].

І лише одна складова MERN відповідає за клієнтську частину сайту – React [25, 22, 30, 23].

Всі роботи по проекту велись в інтегрованому середовищі розробки WebStorm [33]. Для скачування додаткових бібліотек використано менеджер пакетів npm. Для ефективності при роботі зі стилями сторінок було обрано CSS-препроцесор SCSS [29]. Якщо якась чатина коду написана на версіях JavaScript вище ES6 (тобто після 2015 року), то Babel під час збірки проекту здійснює переведення коду на версію ES5, яка зручна браузерам для читання.

Бібліотека Redux [2, 6] використовується у React для контролю станом даних на клієнтській стороні. А JSX допомагає наглядніше поєднати логіку і розмітку в одному файлі (рис. А.1, додаток А) [8].

Теперішні проети зовсім не такі як були 5-10 років тому. Якщо колись статичний сайт на HTML та CSS із додаванням шматочків JavaScript був вершиною у розробці, то тепер сайти динамічні і птребують використання нових методик і технологій. Через значну їх кількість в одному проекті, його структура зазвичай досягає великих розмірів. Файли у проекті мають різні розширеннями та зв’язки. Розробнику стало не під силу готувати такі проекти до завантаження на сервер. Цю роботу виконує Webpack (рис. А.2) [32]. Він займається об’єднанням таких файлів у єдине ціле і має стандартні налаштування, які за необхідності можна змінити.

Як відомо кожна сторінка сайту має внутрішнє представлення у вигляді дерева – DOM (рис. А.3). Існує 12 типів вузлів дерева, але найчастіше ми зустрічаємось із Document (точкою входу), елементами, текстовими вузлами і коментарями. На динамічних сайтах під час виникнення будь-якої події (наприклад, натискання на кнопку) відбувається перемальовування сторінки – змінюється структура DOM-дерева. Рисунок А.4 демонструє особливості потоку даних між вузлами DOM-дерева. Без Redux дані передаються по

принципу «батьківський елемент – дочірній елемент» (ліворуч). Тому передати повідомлення про зміни вузлу, що не знаходиться поруч, досить проблематично. Redux вирішує цю проблему. Кожен вузол має доступ до єдиного джерела істини – Store (праворуч) і тому при передачі повідомлень у будь-яке місце DOM-дерева не викликає значних труднощів

Слід зазначити, що використані технології та засоби розробки веб-сайтів є найпопулярнішими на даний момент і дозволять розробити якісний та швидкий динамічний сайт.

Архітектура веб-сайту.

Веб-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється переважно на сервері, обмін інформацією відбувається по мережі.

Будь-який веб-додаток являє собою набір статичних і динамічних веб-сторінок. Статична веб-сторінка – це сторінка, яка завжди відображається перед користувачем в незмінному вигляді. Веб-сервер відправляє сторінку за запитом веб-браузера без будь-яких змін. На противагу цьому, сервер вносить зміни в динамічну веб-сторінку перед відправкою її браузеру. У зв’язку з тим що сторінка змінюється, вона називається динамічною.

Веб-додаток, розроблений для багатокритеріальної оцінки і вибору піприємств для кредитування, має набір декількох динамічних сторінок, зокрема сторінку для відображення найоптимальнішої альтернативи зі списку підприємств. Тобто, деяка інформація буде визначатися в момент запиту сторінки співробітником.

Коли веб-сервер отримує запит на видачу статичної веб-сторінки, він відправляє сторінку безпосередньо браузеру. Однак, коли запитується динамічна сторінка, дії веб-сервера не настільки однозначні. Сервер передає сторінку спеціальною програмою, яка і формує остаточну сторінку. Така програма називається сервером додатків.

Сервер додатків виконує читання коду, що знаходиться на сторінці, формує остаточну сторінку відповідно до прочитаного коду, а потім видаляє його зі сторінки. В результаті всіх цих операцій виходить статична сторінка, яка передається веб-серверу, який в свою чергу відправляє її клієнтському браузеру. Всі сторінки, які отримує браузер, містять тільки HTML-код. Схематичне зображення процесу показано на рисунку А.5.

Сервер додатків надає можливість використовувати такі ресурси сервера, як бази даних. Наприклад, динамічна сторінка може містити програмні інструкції для сервера додатків, слідуючи яким серверу необхідно отримати певні дані з бази даних і помістити їх в HTML-код сторінки.

Зберігання вмісту в базі даних дозволяє відокремити оформлення веб-сайту від вмісту, яке будуть бачити користувачі. Замість того щоб створювати всі сторінки у вигляді окремих HTML-файлів, пишуться тільки шаблони сторінок для кожного виду інформації, що представляється. Потім вміст завантажується в базу даних, після чого веб-сайт буде витягувати його при запитах користувачів. Крім того, можна оновити інформацію в одному джерелі і продублювати це зміна на всіх веб-сайті без редагування кожної сторінки вручну (рис. А.6).

Сервер додатків не може безпосередньо отримати дані з бази, оскільки бази даних використовують специфічні формати зберігання даних. Тому для підключення до баз даних сервер додатків використовує посередника – драйвер бази даних. Він являє собою програмний модуль, за допомогою якого встановлюється взаємодія між сервером додатків і базою даних.

Після того як драйвер встановить з’єднання, виконується запит до бази, в результаті чого формується набір записів. Набір записів повертається до сервера додатків, який використовує отримані дані для формування сторінки.

Для використання в веб-додатку придатна будь-яка база даних за умови, що на сервері встановлений відповідний драйвер бази даних.

Отже, розроблений у випускному кваліфікаційному проекті сайт є веб-додатком з динамічними веб-сторінками, оскільки вони формуються веб-

сервером у відповідь на запити користувача, сформовані під час користування сторінкою.

3.2. Алгоритмічна реалізація методів у сфері банківського кредитування

Загальний алгоритм роботи методів показано на рисунку 3.2.

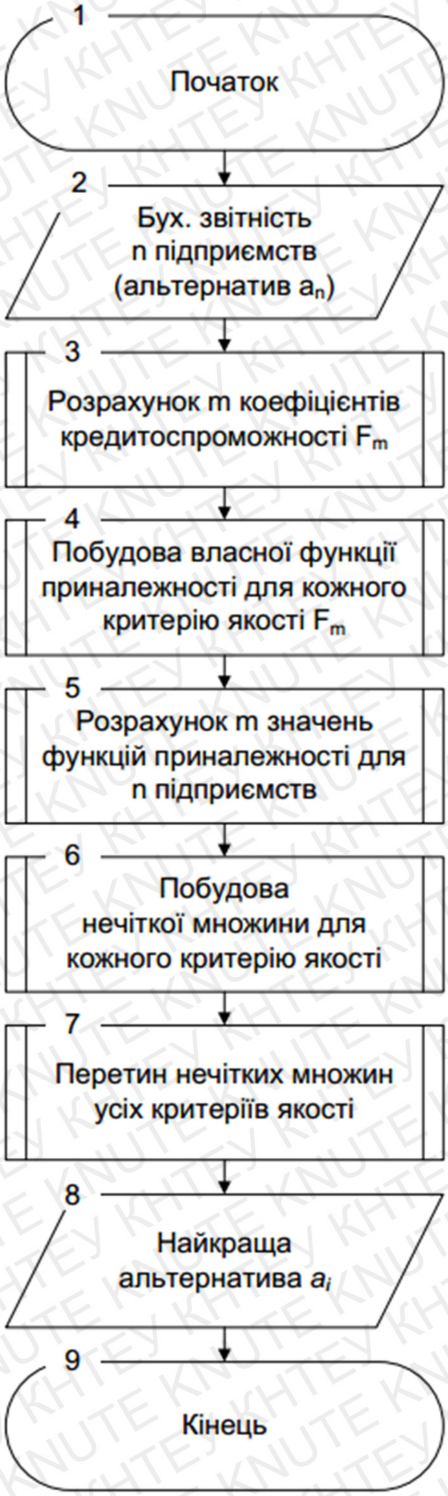

Рис. 3.2. Доступ до бази даних

Уся робота алгоритму реалізована у системі Mathcad для наглядності усього процесу. Для прикладу було взято 4 альтернативи (рис. 3.3) (значення абревіатур у підрозділі 2.3). Розрахунок коефіцієнтів кредитоспроможності показано на рисунках 3.4 та 3.5

ГС := (229.1 946.2 947 1442.9)

КФВ := (394.1 462.7 466.4 2066)

ДЗ := (4639.8 8391.4 8514.5 10908.2)

ЗВ := (6028.1 21557.6 21370.4 17424.5)

ВК := (12395.8 35247.8 41244.2 53939.4)

ЗКс := (4058.1 13834.9 16827.1 25028.3)

ПБ := (16453.9 49082.7 58071.3 78967.7)

ВВ := (59438.9 38567.9 43589.5 28343.6)

П := (16642.9 4442.5 65384.2 3401.2)

Рис. 3.3. Дані бухгалтерської звітності для 4-х альтернатив

$$i := 1..5 \quad j := 1..4$$

$$F_{i,j} := \begin{pmatrix} \frac{ГС_{1,j} + КФВ_{1,j}}{ЗКс_{1,j}} \\ \frac{ГС_{1,j} + КФВ_{1,j} + ДЗ_{1,j}}{ЗКс_{1,j}} \\ \frac{ГС_{1,j} + КФВ_{1,j} + ДЗ_{1,j} + ЗВ_{1,j}}{ЗКс_{1,j}} \\ \frac{ВК_{1,j}}{ПБ_{1,j}} \\ \frac{П_{1,j}}{ВВ_{1,j}} \end{pmatrix}_i$$

Рис. 3.4. Формули для оцінки критеріїв якості кожної альтернативи

У блок-схемі (рис. 3.8) 4-й блок реалізовано шляхом побудови 5-ти функцій приналежності. В якості числових параметрів *a, b* обрано довільні дійсні значення, що впорядковані відношенням *a<b* (рис. Б.1 – Б.5). Межі для осі OX вибрано таким чином, щоб добре проглядались нормативні значення критерію (підрозділ 2.3).

$$F = \begin{pmatrix} 0.154 & 0.102 & 0.084 & 0.14 \\ 1.297 & 0.708 & 0.59 & 0.576 \\ 2.782 & 2.267 & 1.86 & 1.272 \\ 0.753 & 0.718 & 0.71 & 0.683 \\ 0.28 & 0.115 & 1.5 & 0.12 \end{pmatrix}$$

Рис. 3.5. Результати обчислення коефіцієнтів

Реалізація блоку 5 блок-схеми відображена на рисунку 3.6. На основі отриманих результатів будуються нечіткі множини для кожного критерію (блок 6 блок-схеми).

$$G(x) := (F1(x)\ \ F2(x)\ \ F3(x)\ \ F4(x)\ \ F5(x))$$

$$i := 1..5 \qquad j := 1..4$$

$$\mu_{i,j} := G\left(F_{i,j}\right)1,i$$

$$\mu = \begin{pmatrix} 0.295 & 0.13 & 0.088 & 0.246 \\ 0.987 & 0.509 & 0.378 & 0.363 \\ 0.923 & 0.783 & 0.639 & 0.411 \\ 0.992 & 0.974 & 0.969 & 0.948 \\ 0.109 & 0.069 & 0.227 & 0.07 \end{pmatrix}$$

Рис. 3.6. Обчислення значення функції для усіх критеріїв

Останній етап – згортка (рис. 3.7).

$$i := 1..5 \qquad j := 1..4$$

$$B_{1,j} := \min\left(\mu^{\langle j \rangle}\right) \qquad B = (0.109\ \ 0.069\ \ 0.088\ \ 0.07)$$

$$Result := \max(B)$$

$$Result = 0.109$$

Рис. 3.7. Згортка та виявлення кращої альтернативи

Немає конкретної прив’язки до форми функцій приналежності критеріїв – її визначає експерт. Тому значення результуючого вектора пріоритетів альтернатив відрізняється від значень, зазначених у підрозділі 2.3 у з’явзку із побудовою функцій приналежності дещо іншої форми.

Отже, найкращою альтернативою є a_1, якій відповідає значення 0.295. На другому, третьому і четвертому місцях знаходяться відповідно $a_4 \rightarrow 0.246$, $a_2 \rightarrow 0.130$, $a_3 \rightarrow 0.088$.

3.3. Програмна реалізація КСВПК

Існують набори інструментів React, які допомагають для вирішення таких завдань як:

- масштабування до великої кількості файлів та компонентів;
- використання сторонніх бібліотек із npm;
- раннє виявлення поширених помилок;
- відображення змін CSS і JS на льоту в процесі розробки;
- оптимізація коду для production.

Create React App – зручне середовище для вивчення React і кращий спосіб почати створення нового односторінкового додатку на React.

Інструмент налаштовує середовище для використання нових можливостей JavaScript, оптимізує додаток для production і забезпечує комфорт під час розробки. Потрібен лише Node.js не нижче версії 8.10 і npm не нижче версії 5.6.

Для створення проекту потрібно виконати команди:

```
npx create-react-app book-store
cd book-store
npm start
```

В результаті буде створено папку book-store із початковим набором файлів. Структура проекту зображена на рисунку 3.8.

npx у першому рядку не є помилкою. Це інструмент запуску пакетів, доступний у версіях npm 5.2 і вище.

npx – інструмент, призначений для того, щоб допомогти стандартизувати досвід використання npm-пакетів: так само, як і npm спрощує установку та управління залежностями, розміщеними в реєстрі, npx спрощує використання CLI-утиліт та інших виконуваних файлів. Це значно спрощує ряд речей, які робились раніше за допомогою звичайного npm.

Create React App не обробляє back-end-логіку або бази даних. Він тільки надає команди для збирання front-end, тому можна використовувати його із будь-яким back-end. Create React App включає Babel та webpack із попередніми налаштуваннями, що є дуже зручно.

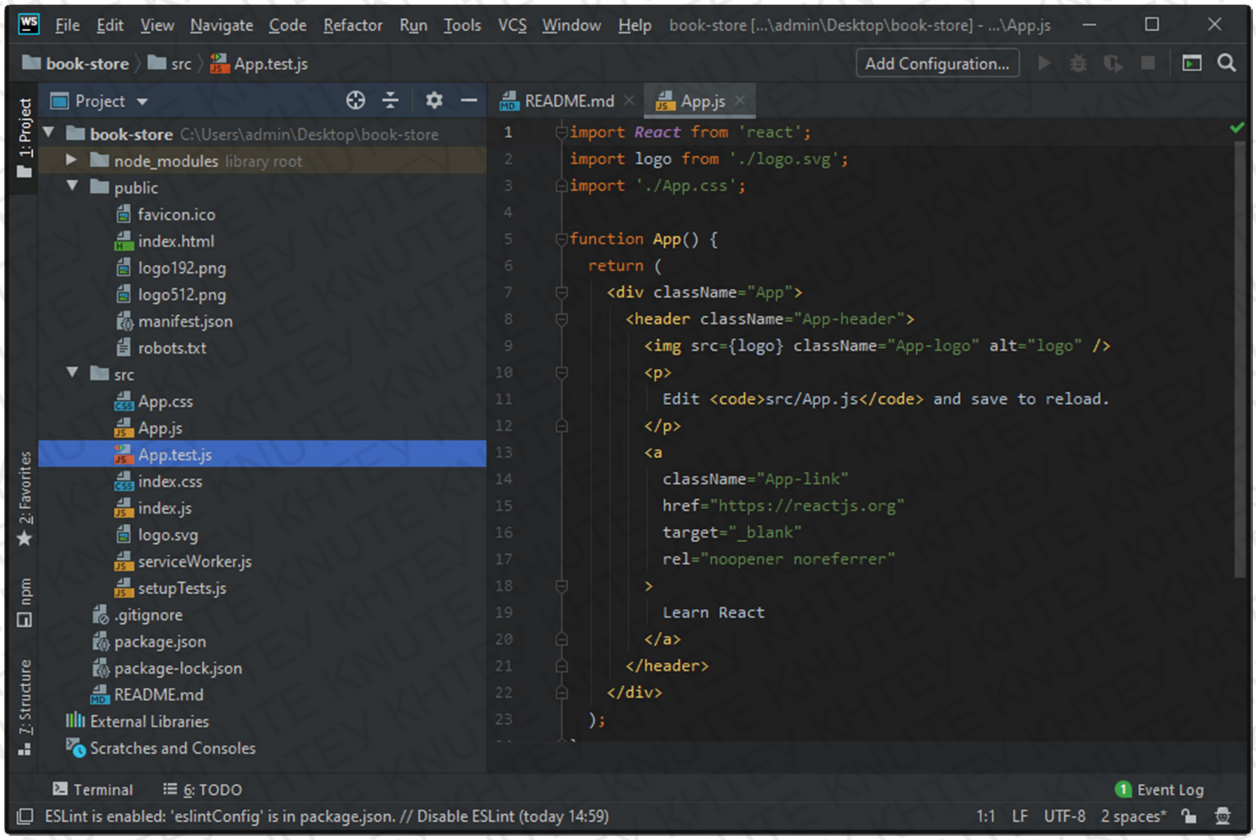

Рис. 3.8. Структура файлів у новому проекті

Для запуску сервера на локальному хості 3000 у командній стрічці потрібно виконати команду `npm start`. З'явиться повідомлення, що сайт запущений на локальному хості за адресою http://localhost:3000 (рис. 3.20). Коли додаток готовий до розгортання в production, запуск команди npm run build створить оптимізовану збірку застосунку в папці build.

Отже, разом з Create React App було отримано готовий, налаштований сервер, Babel-інтерпретатор, webpack для збірки, засоби для написання css-коду без префіксів для кросбраузерності та налаштоване середовище для тестування і збирання проекту.

У середовищі буде все, що потрібно для створення сучасного React-додатку:

- React, JSX і ES6;

- додаткові можливості мови за межами ES5: деструктуризація, стрілкові функції, шаблони стрічок, проміси, підтримка асинхронності (async/await), спосіб створення змінних через let та const;

- dev-сервер, який працює незалежно від незначних помилок;

- можливість імпортувати CSS-файли та зображення безпосередньо з JavaScript;

- autoprefixer CSS, що дозволяє відмовитись від -webkit та інших префіксів;

- сценарій збірки для об'єднання JS, CSS та зображень для production за допомогою sourcemaps.

Структура готового додатку показана на рисунку 3.9.

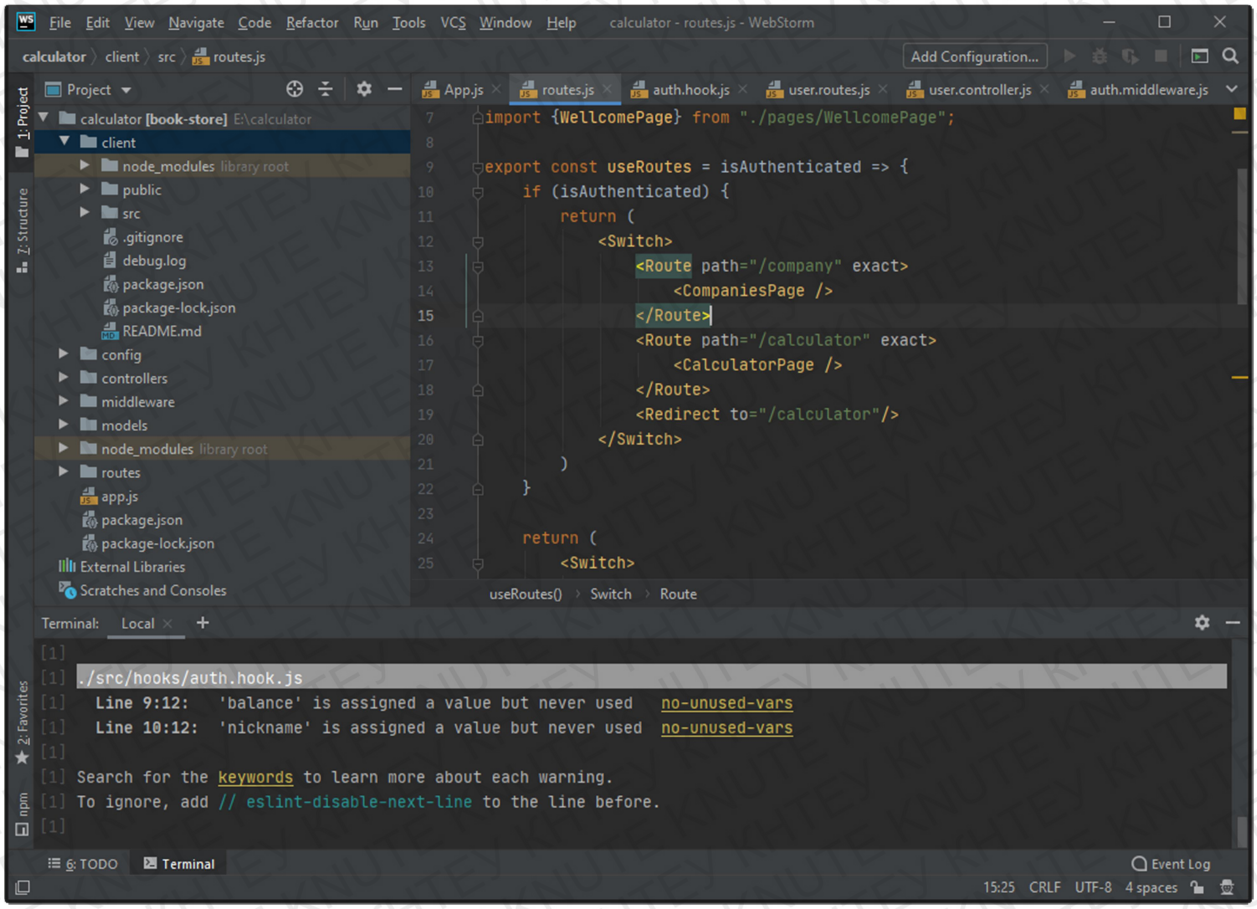

Рис. 3.9. Структура проекту на фінальній стадії

Web-сайт містить вітальну сторінку, сторінку авторизації, сторінку зі списком усіх компанії та, власне, сторінку, на якій відбувається обрахунок на основі даних вибраних компаній (рис. 3.10-3.13).

Авторизація необхідна для запобігання пошкодження даних у базі.

Інформація про підприємства зберігається у базі даних, підключення до якої відбувається на стороні сервера. Форма збереження даних відображена у додатку В.

Програмний код сайту представлено у додатках Г, Д та Е.

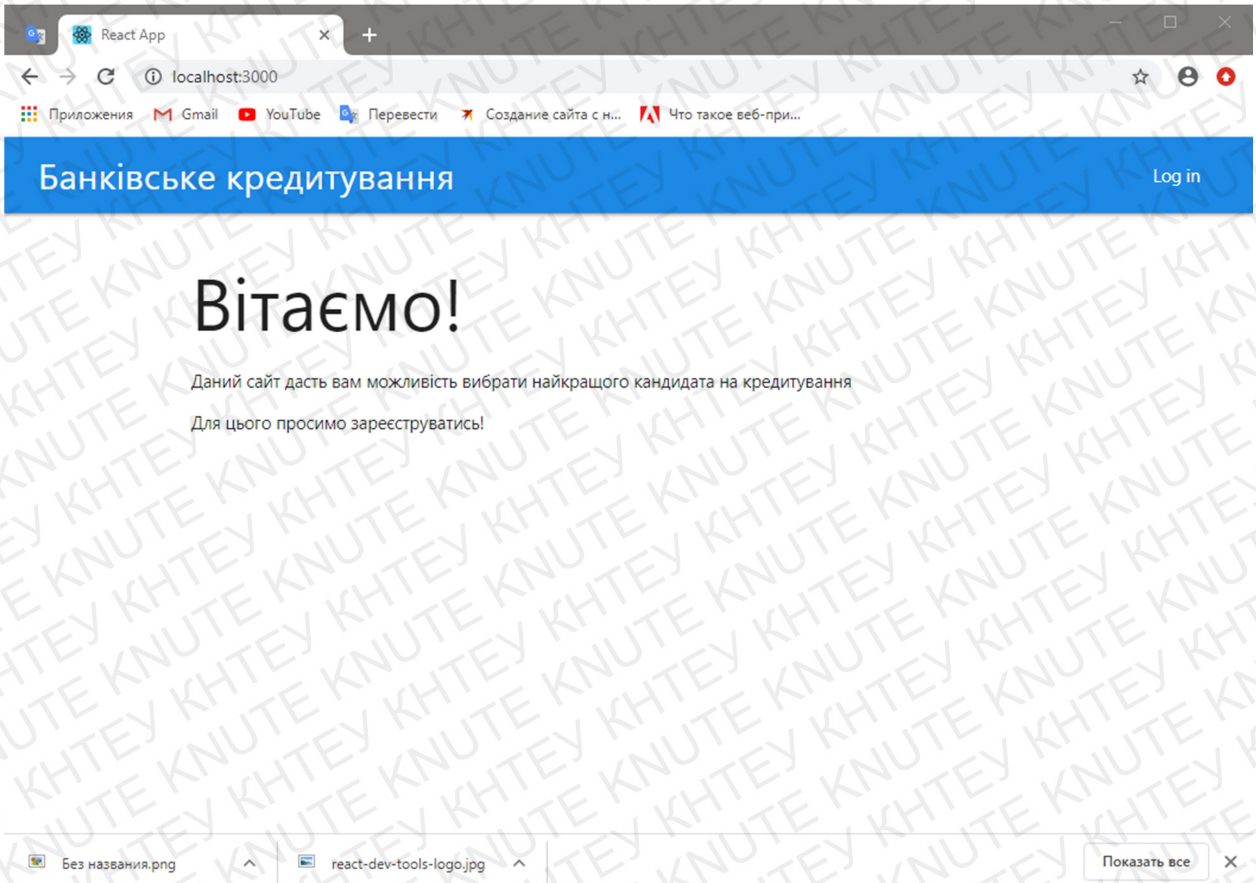

Рис. 3.10. Вітальна сторінка

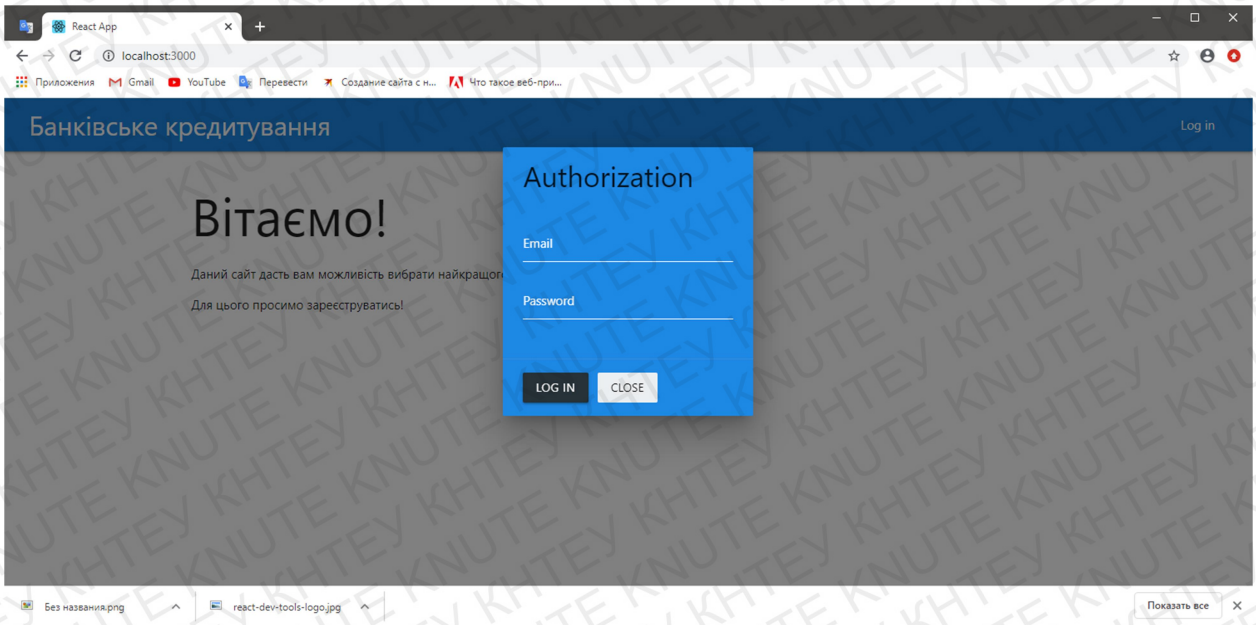

Рис. 3.11. Сторінка авторизації

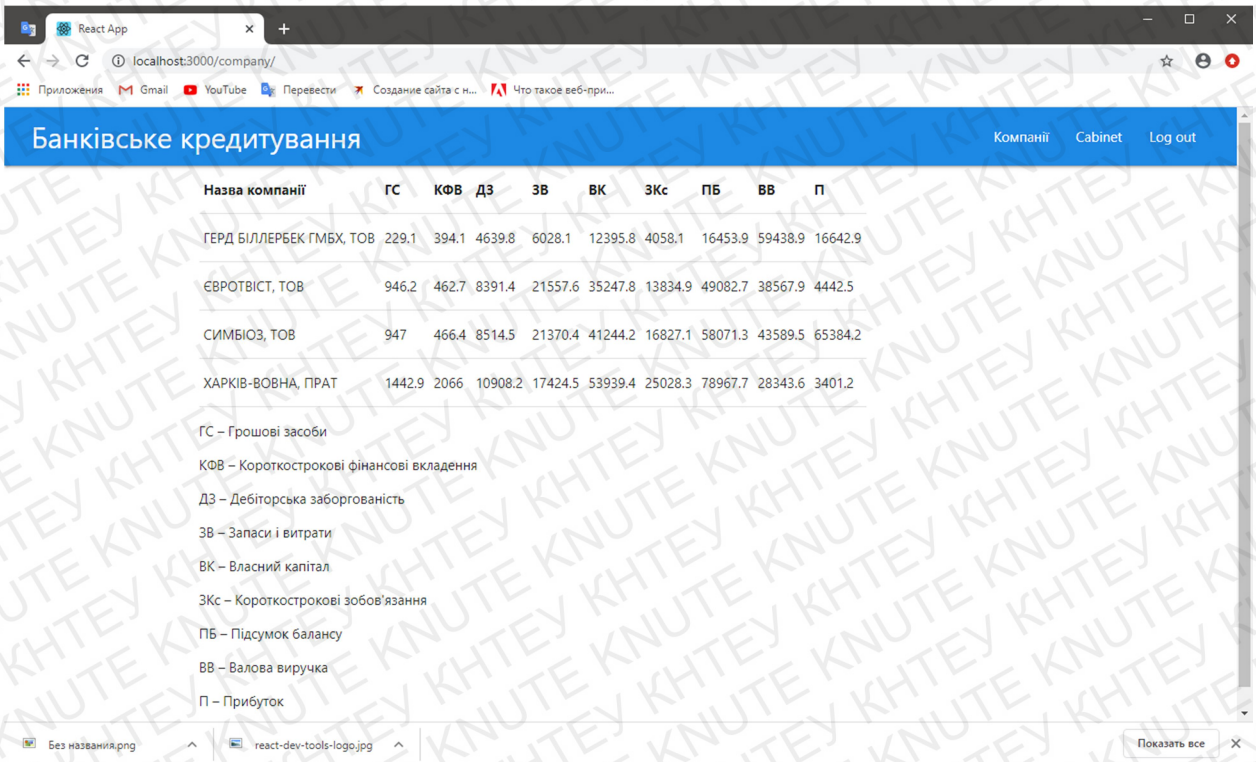

Рис. 3.12. Сторінка зі списком усіх компаній

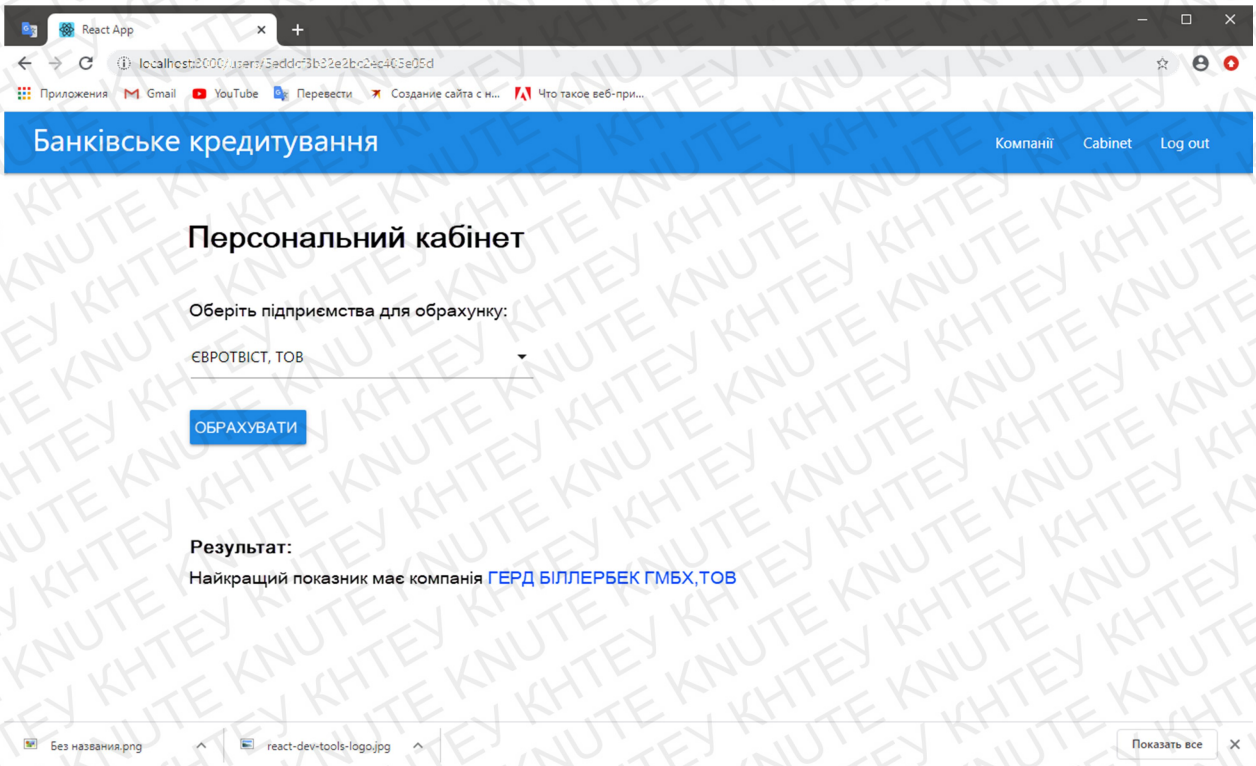

Рис. 3.13. Сторінка обрахунку найкращої альтернативи зі списком підприємств із множинним вибором

ВИСНОВКИ

У випускному кваліфікаційному проекті представлено результати теоретичних і прикладних досліджень, що полягають у розробці інформаційної системи, яка призначена для багатокритеріальної оцінки та відбору підприємств для кредитування. Результатом досліджень стала розробка web-сайту за допомогою сучасних технологій та методів.

В результаті проведених досліджень було отримано такі результати:

- Банки сприяють розвитку економіки за допомогою їх пріоритетної функції – кредитування, що дозволяє забезпечити швидкий і цивілізований розвиток підприємницької діяльності на внутрішньому і зовнішньому економічному просторі. Проте кожен банк повинен мати апарат розрахунку усіх ризиків, щоб максимально убезпечити себе від збитків і отримати прибуток.

- Розроблено аналітичний апарат розрахунку коефіцієнтів кредитоспроможності позичальків

- Обрано метод максимінної згортки, який на базі теорії нечітких множин дозволяє отримати кращу альтернативу із представлених.

- Здійснено огляд усіх функцій приналежності та методів їх побудови і обрано ті, які дозволяють краще відобразити характер фінансових показників.

- Розроблено сайт, у якому закладено розроблений алгоритм багатокритеріальної оцінки підприємств на кредитоспроможність. У розробці використано найсучасніші технології (React, Node.js, MongoDD, Express), що дозволяють створити якісний динамічний сайт. На сайті присутня авторизація. Є можливість видаляти / редагувати / додавати нові підприємства та критерії відбору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chuba N. npm vs Yarn – какой менеджер пакетов стоит использовать? [Електронний ресурс] / Nazar Chuba. – Електрон. дані. – Режим доступу: https://ua-blog.com/npm-vs-yarn-npm-vs-yarn-какой-менеджер-паке-тов-стоит-испол/. – Назва з екрану. – Дата публікації: 29.03.2018. – Дата перегляду: 30.03.2020.

2. Redux-in-russian [Електронний ресурс]: оригинальная документация по Redux с переводом на руський. – Електрон. дані. – Режим доступу: https://rajdee.gitbooks.io/redux-in-russian/content/. – Дата публікації: 16.10.2018. – Дата перегляду: 03.04.2020.

3. Автоматизированные информационные технологии в экономике: Учебник / Под ред. Г. А. Титоренко. – М.: Компьютер, ЮНИТИ, 1998. – 400 с.

4. Андрейчиков А.В., Андрейчикова О.Н. Анализ, синтез, планирование решений в экономике – М.: Финансы и статистика, 2000. – 368 с.

5. Борисов А. П., Крумберг О. А., Федоров И. П. Принятие решений на основе нечетких моделей. – Рига: Зинатне, 1990. – 184 с.

6. Бэнкс А. React и Redux: функциональная веб-разработка / А. Бэнкс, Е. Порселло. – СПб.: Питер, 2018. – 336 с.

7. Варналій З. С. Мікрокредитування як чинник розвитку малого підприємництва / Захарій Степанович Варналій // Вісник КНТЕУ. – 2007. – № 4.

8. Введение в JSX [Електронний ресурс]. – Електрон. дані. – Режим доступу: https://thewebland.net/development/javascript/reactjs/introducing-jsx/. – Дата публікації: 06.11.2017. – Дата перегляду: 01.04.2020.

9. Волошин І. Оптимальне управління роздрібним кредитуванням банку / І. Волошин // Вісник НБУ. – 2010. – № 5.

10. Глущенко С. В. Напрямки кредитування суб'єктів малого бізнесу в Україні / С.В. Глущенко // Науково-практичний журнал Національного банку України. – 2005. – № 4. – С. 81-93.

11. Дмитров С. Система скорингу на основі індикаторів ризику як ефективна складова фінансового моніторингу в банку / С. Дмитров, В. Черняк, О. Кузьменко // Вісник НБУ. – 2011. – № 1.

12. Ендовицкий Д.А. Анализ кредитоспособности организации и группы компаний: [учебное пособие] / К.В. Бахтин, Д.В. Ковтун. – КноРус, 2012. – 375 с.

13. Жариков В.В. Управление кредитными рисками. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2009. – 244с.

14. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений: Пер. с англ. – М.: Мир, 1976. – 165 с.

15. Исаев Р. А. Банковский менеджмент и бизнес-инжиниринг: в 2 т. Т. 2. / Р. А. Исаев. – М.: Инфра-М, 2015. – 336 с.

16. Карпова О.І. Особливості організації мікрокредитування малого та середнього бізнесу / О. І. Карпова, Л. І. Чубарєва // Финансы, учет, банки: [Сборник научных трудов ДонНУ]. – 2012. – Выпуск №1 (18). – С. 145-153.

17. Коцовська Р. Кредитна підтримка малого та середнього бізнесу установами фінансового ринку України / Р. Коцовська // Регіональна економіка. – 2002. – № 3.

18. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. – СПб.: БХВ Петербург, 2005. – 736 с.

19. Масленченков Ю.С. Работа банка с корпоративными клиентами / Ю.С. Масленченков, Ю.Н. Тронин. – М.: Юнити-Дана, 2011. – 358с.

20. Нечеткие множества в моделях управления и искусственного интел-лекта / под ред. Д. А. Поспелова. – М.: Наука, 1986. – 312 с.

21. Нечеткие множества и теория возможностей. Последние достиже-ния: Пер. с англ. – М.: Радио и связь, 1986. – 408 с.

22. Структура файлов [Електронний ресурс]: есть ли рекомендации по структуре React-проектов? / Документация. – Електрон. дані. – Режим доступу: https://ru.reactjs.org/docs/faq-structure.html/. – Дата публікації: 19.03.2020. – Дата перегляду: 05.04.2020.

23. Тиленс Томас Марк. React в действии / Томас Марк Тиленс. – СПб.: Питер, 2019. – 368 с.

24. Финансы предприятий: Учебник / Руководитель авт. кол. и науч. ред. проф. А. М. Поддерьогин. 3-е изд., перераб. и доп. – К.: КНЭУ, 2001. – 460 с.

25. Фоменко А. Что лучше всего изучать в 2020 году: Angular, React или Vue.js [Електронний ресурс] / Александр Фоменко // JAVASCRIPT. – Електрон. дані. – Режим доступу: https://badcode.ru/chto-luchshie-vsiegho-izuchat-v-2020-ghodu-angular-react-ili-vue-js/. – Назва з екрану. – Дата публікації: 29.02.2020. – Дата перегляду: 20.03.2020.

26. Фурдак В. Как стать full stack разработчиком, зная back-end. Пошаговая инструкция [Електронний ресурс] / Владислав Фурдак. – Електрон. дані. – Режим доступу: https://dou.ua/lenta/articles/how-to-become-full-stack-developer/. – Назва з екрану. – Дата публікації: 27.03.2019. – Дата перегляду: 31.03.2020.

27. La Banque et le risque PME / G. Chanel-Reynaud, É. Bloy, J.-P. Allégret et autres; sous la direction de G. Chanel-Reynaud et É. Bloy. – Lyon: Presses Universitaires de Lyon, 2001. – P.5.

28. Guarantees and Mutual Guarantees / BEST Report. Report to the Commission by an Independent Expert Group. – Електрон. дані. – Режим доступу: http://ec.europa.eu/enterprise/entrepreneurship/financing/docs/guarantees_best_ report.pdf – Назва з екрану. – Дата публікації: 11.01.2015. – Дата перегляду: 28.02.2020

29. Kryukov D. Sass vs Less: Which CSS Preprocessor to Choose in 2019 [Електронний ресурс] / Denis Kryukov. – Електрон. дані. – Режим доступу:

https://blog.soshace.com/sass-vs-less-which-css-preprocessor-to-choose-in-2019/. – Назва з екрану. – Дата публікації: 02.07.2019. – Дата перегляду: 28.03.2020.

30. Marcus I. Easy React JS for Beginner Developers: A Step-By-step Visual Guide to Learn React Js and Building Your Own React Applications from Scratch / Ibas Marcus. – Independently Published, 2019. – 202 c.

31. Mendel J.M. Uncertain rule-based fuzzy logic systems: introduction and new directions. Prentice-Hall, PTR, Upper Saddle River. – NJ, 2001. – 555 p.

32. Nandan Singh A. An intro to Webpack: what it is and how to use it [Електронний ресурс] / Ashish Nandan Singh. – Електрон. дані. – Режим доступу: https://www.freecodecamp.org/news/an-intro-to-webpack-what-it-is-and-how-to-use-it-8304ecdc3c60/. – Дата публікації: 15.01.2019. – Дата перегляду: 04.04.2020.

33. Yusov K. Top 15 JavaScript IDEs and JS Editors for Frontend Development [Електронний ресурс] / Kirill Yusov. – Електрон. дані. – Режим доступу: https://jelvix.com/blog/best-javascript-ides/. – Назва з екрану. – Дата публікації: 15.01.2020. – Дата перегляду: 29.03.2020.

ДОДАТКИ

Додаток А

Технології, на базі яких побудовано web-сайт

Рис. А.1. JS vs JSX-синтаксис

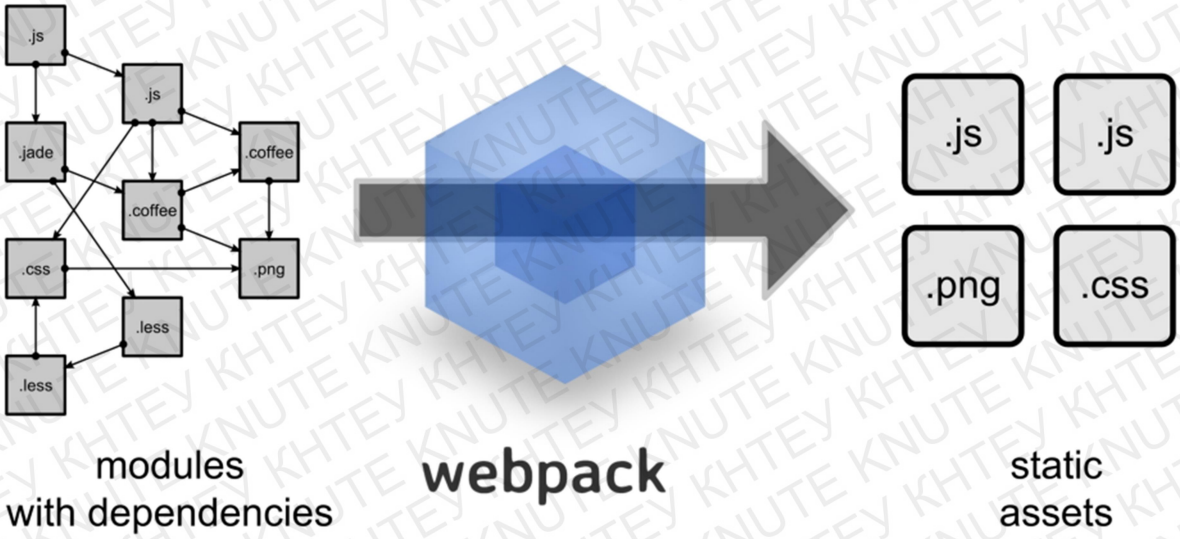

Рис. А.2. Суть роботи збирача проектів Webpack

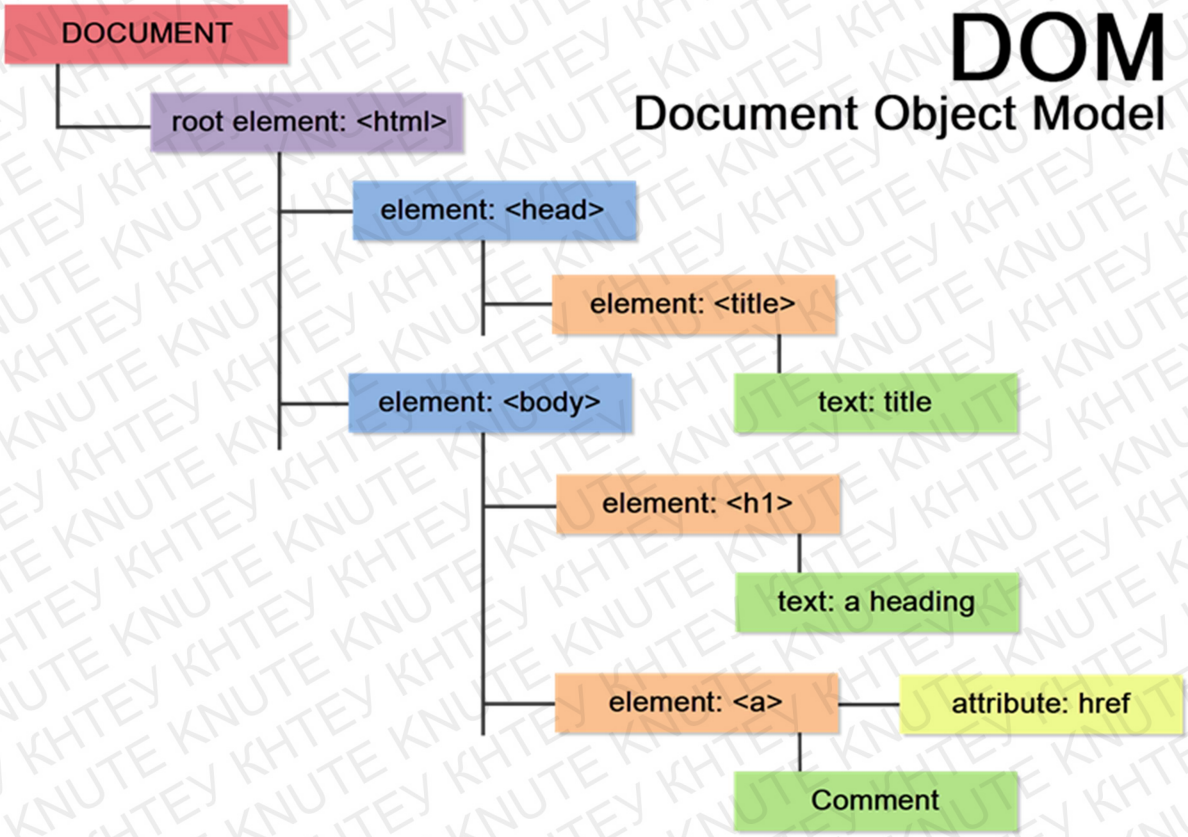

Рис. А.3. Деревовидна структура HTML-сторінки

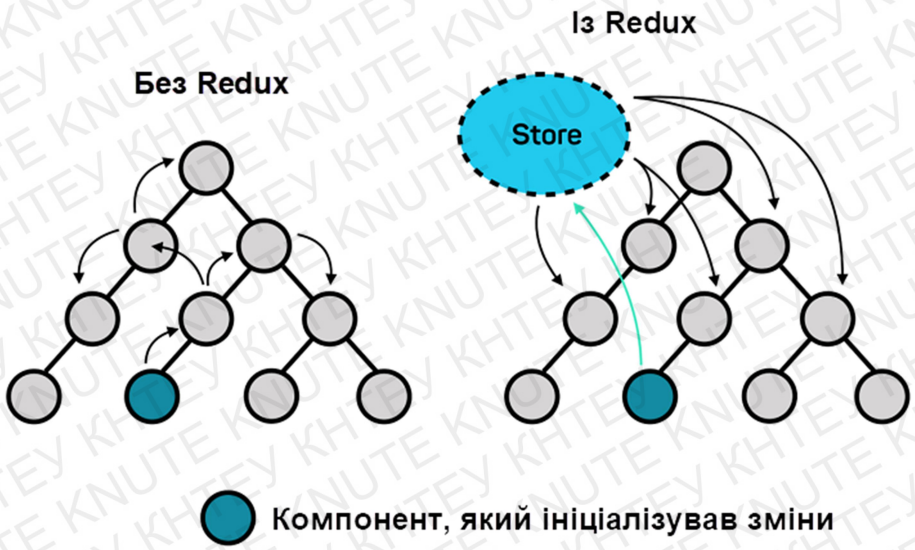

Рис. А.4. Способи передачі даних між вузлами DOM-дерева

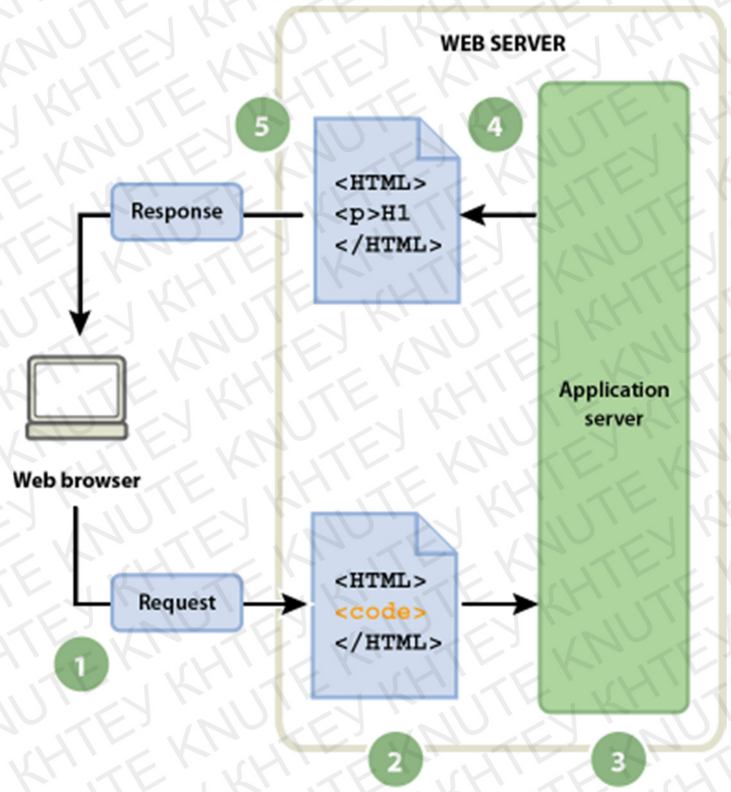

Рис. А.5. Обробка динамічних сторінок

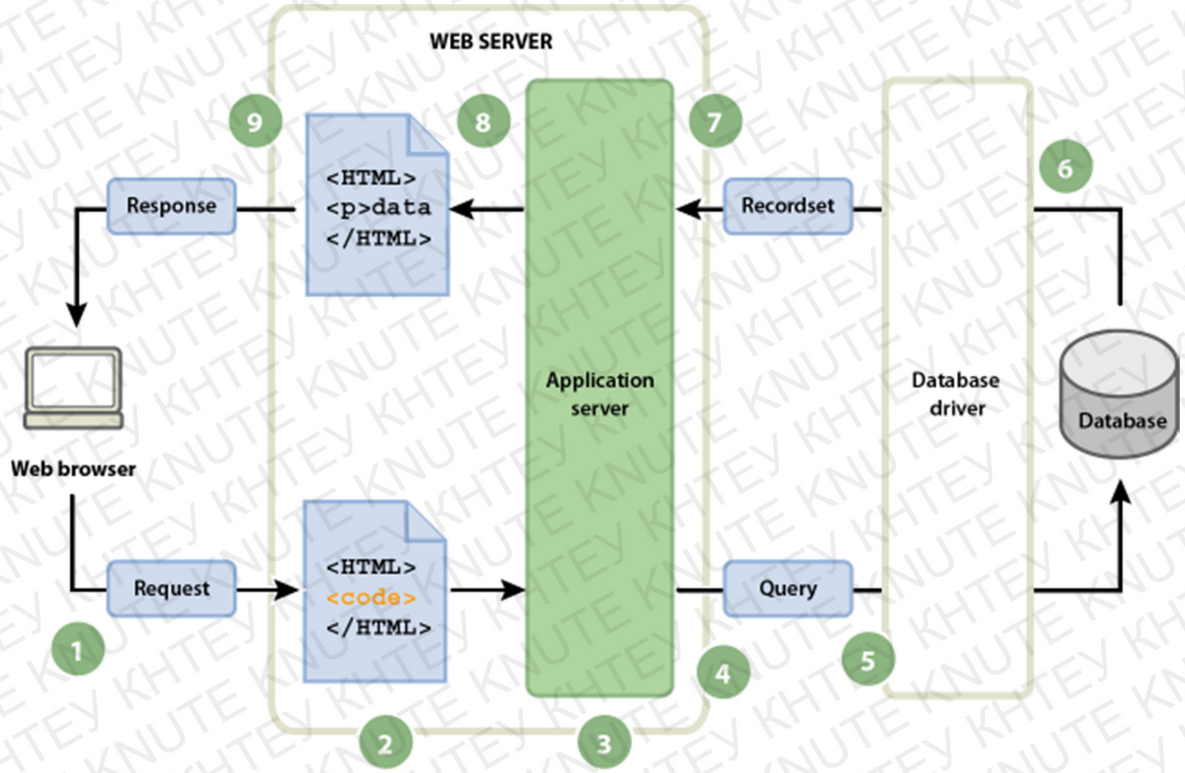

Рис. А.6. Доступ до бази даних

Додаток Б

Функції приналежності

Критерій якості **F1**

Функція приналежності класу S (тип 1)

$a := 0 \qquad b := 0.4$

$$F1(x) := \begin{cases} 0 & \text{if } x \le a \\ 2\cdot\left(\frac{x-a}{b-a}\right)^2 & \text{if } a < x \le \frac{a+b}{2} \\ 1-2\cdot\left(\frac{b-x}{b-a}\right)^2 & \text{if } \frac{a+b}{2} < x < b \\ 1 & \text{if } b \le x \end{cases}$$

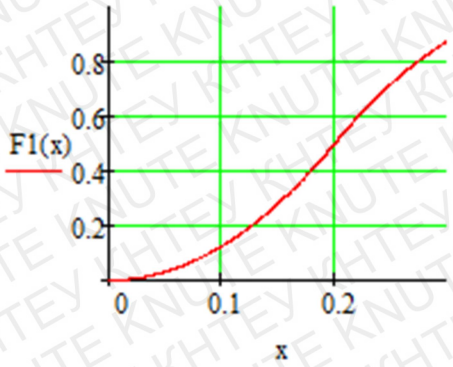

Рис. Б.1. Функція приналежності для критерію якості F_1

Критерій якості **F2**

Функція приналежності класу S (тип 2)

$a := 0 \qquad b := 1.4$

$$F2(x) := \begin{cases} 0 & \text{if } x < a \\ \frac{1}{2}+\frac{1}{2}\cdot\cos\left(\frac{x-b}{b-a}\cdot\pi\right) & \text{if } a \le x \le b \\ 1 & \text{if } x > b \end{cases}$$

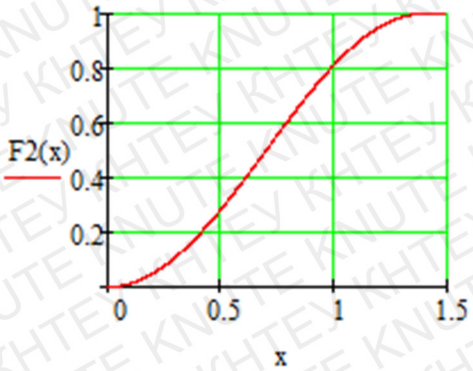

Рис. Б.2. Функція приналежності для критерію якості F_2

Критерій якості **F3**

Функція приналежності класу S (тип 2)

$a := -0.5 \qquad b := 3.5$

$$F3(x) := \begin{cases} 0 & \text{if } x < a \\ \frac{1}{2} + \frac{1}{2}\cdot\cos\left(\frac{x-b}{b-a}\cdot\pi\right) & \text{if } a \le x \le b \\ 1 & \text{if } x > b \end{cases}$$

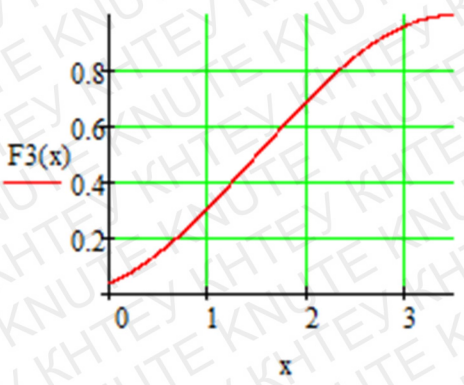

Рис. Б.3. Функція приналежності для критерію якості F_3

Критерій якості **F4**

Функція приналежності класу S (тип 1)

$a := 0 \qquad b := 0.8$

$$F4(x) := \begin{cases} 0 & \text{if } x < a \\ \frac{1}{2} + \frac{1}{2}\cdot\cos\left(\frac{x-b}{b-a}\cdot\pi\right) & \text{if } a \le x \le b \\ 1 & \text{if } x > b \end{cases}$$

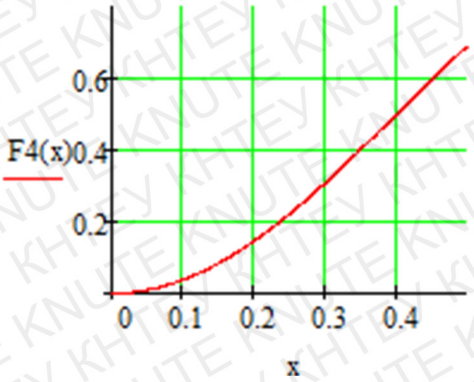

Рис. Б.4. Функція приналежності для критерію якості F_4

Критерій якості **F5**

П-подібна функція приналежності (дзвіноподібна)

$a := 0.3 \quad b := 1.2 \quad c := 1$

$$F5(x) := \frac{1}{1 + \left(\left| \frac{x - c}{a} \right| \right)^{2 \cdot b}}$$

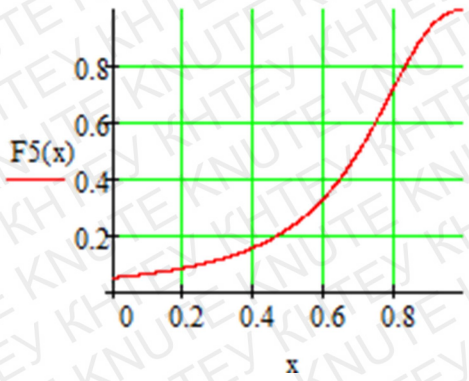

Рис. Б.5. Функція приналежності для критерію якості F_5

Додаток В

Робота з базою даних MongoDB

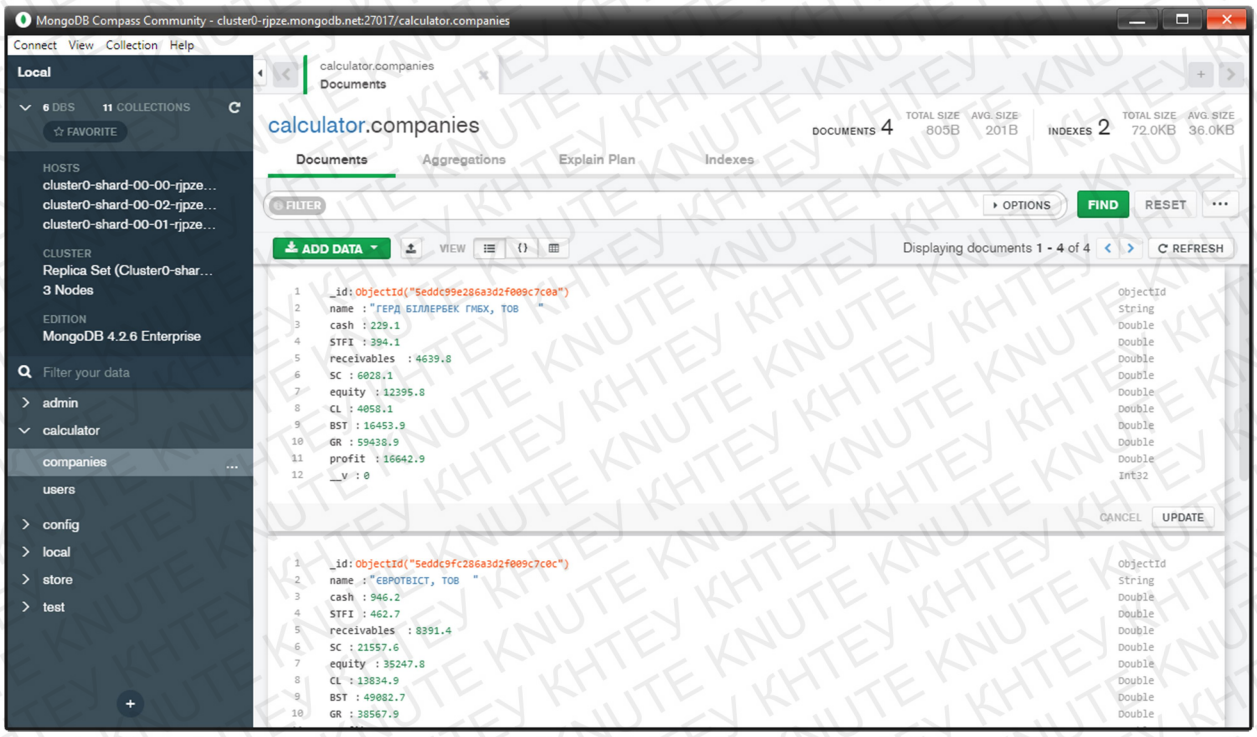

Рис. В.1. База даних із інформацією про підприємства

Додаток Г

Структура проекту сайту

Рис. Г.1. Структура web-сайту

Додаток Д

Front-end

Публічні файли

public / index.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta
   name="description"
   content="Web site created using create-react-app"
  />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <!--
   manifest.json provides metadata used when your web app is installed on a
   user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
  -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
   Notice the use of %PUBLIC_URL% in the tags above.
   It will be replaced with the URL of the `public` folder during the build.
   Only files inside the `public` folder can be referenced from the HTML.

   Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
   work correctly both with client-side routing and a non-root public URL.
   Learn how to configure a non-root public URL by running `npm run build`.
  -->
  <title>React App</title>
 </head>
 <body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
   This HTML file is a template.
   If you open it directly in the browser, you will see an empty page.

   You can add webfonts, meta tags, or analytics to this file.
   The build step will place the bundled scripts into the <body> tag.

   To begin the development, run `npm start` or `yarn start`.
   To create a production bundle, use `npm run build` or `yarn build`.
  -->
 </body>
</html>
```

Компонент Auth

client / src / components / Auth / Auth.js

```
import React, {useState} from 'react'
import {FormContext} from "../../context/FormContext";
import {LoginBtn} from "../LoginBtn";
import {Button} from "react-materialize"
import './Auth.scss'


export const Auth = () => {

  const [form, setForm] = useState({nickname: ''})

  const changeHandler = event => {
    setForm({...form, [event.target.name]: event.target.value})
  }


  return (
    <FormContext.Provider value={[form, setForm]} >
      <div id="login">
        <div className="card blue darken-1">
          <div className="card-content white-text">
            <div className="input-field">
              <input
                id="email"
                type="email"
                className="validate yellow-input"
                name="email"
                onChange = {changeHandler}
                autoComplete = "off"
              />
              <label htmlFor="email">Email</label>
            </div>
            <div className="input-field">
              <input
                id="password"
                type="password"
                className="validate yellow-input"
                name="password"
                onChange = {changeHandler}
                autoComplete = "off"
              />
              <label htmlFor="nickname">Password</label>
            </div>
          </div>
          <div className="card-action">
            <LoginBtn />
            <Button
              flat
              modal="close"
              node="button"
              waves="green"
              className="blue-grey lighten-5 black-text"
            >
              Close
            </Button>
```

```
            </div>
          </div>
        </div>
      </FormContext.Provider>
    )
}
```

client / src / components / Auth / Auth.scss

```
.card {
  margin: 0;
  box-shadow: none;
}
.card-action button {
  margin-right: 10px;
}

#login input {
  border-bottom-color: #fff;
}

#login input:hover,
#login input:focus,
#login input:active {
  border-bottom-color: #fff;
  box-shadow: 0 1px 0 0 #fff;
  color: #fff;
}
input.valid:not(.browser-default) {
  box-shadow: 0 1px 0 0 #fff!important;
  color: #fff;
}

#login input + label,
#login input:hover + label,
#login input:focus + label,
#login input:active + label {
  color: #fff;
}
```

Компонент CompaniesList

client / src / components / CompaniesList/ CompaniesList.js

```
import React from 'react'
import './CompaniesList.scss'

export const CompaniesList = ({companies}) => {

  if (!companies) {
    return <p className="center">Компанії наразі відсутні</p>
  }

  return (
    <div className="row">
      <div className="col col9">
        <table>
          <thead>
          <tr>
            <th>Назва компанії </th>
            <th>ГС</th>
            <th>КФВ</th>
```

```
                <th>ДЗ</th>
                <th>ЗВ</th>
                <th>ВК</th>
                <th>ЗКс</th>
                <th>ПБ</th>
                <th>ВВ</th>
                <th>П</th>
              </tr>
              </thead>

              <tbody>
                {
                  companies.map(company => {
                    return (
                      <tr>
                        <td>{company.name}</td>
                        <td>{company.cash}</td>
                        <td>{company.STFI}</td>
                        <td>{company.receivables}</td>
                        <td>{company.SC}</td>
                        <td>{company.equity}</td>
                        <td>{company.CL}</td>
                        <td>{company.BST}</td>
                        <td>{company.GR}</td>
                        <td>{company.profit}</td>
                      </tr>
                    )
                  })
                }
              </tbody>
            </table>
          </div>
          <div className="col col3">
            <div>
              <p>ГС – Грошові засоби</p>
              <p>КФВ – Короткострокові фінансові вкладення </p>
              <p>ДЗ – Дебіторська заборгованість  </p>
              <p>ЗВ – Запаси і витрати  </p>
              <p>ВК – Власний капітал  </p>
              <p>ЗКс – Короткострокові зобов'язання</p>
              <p>ПБ – Підсумок балансу </p>
              <p>ВВ – Валова виручка</p>
              <p>П – Прибуток </p>
            </div>
          </div>
        </div>

    )
}
```

Компонент Loader

client / src / components / Loader/ Loader.js

```
import React from 'react'
import './Loader.scss'

export const Loader = () => {
  return (
    <div id="loader">
      <div className="preloader-wrapper big active">
        <div className="spinner-layer spinner-blue-only">
          <div className="circle-clipper left">
```

```
            <div className="circle"/>
          </div>
          <div className="gap-patch">
            <div className="circle"/>
          </div>
          <div className="circle-clipper right">
            <div className="circle"/>
          </div>
        </div>
      </div>
    </div>
  )
}
```

client / src / components / Loader/ Loader.scss

```
#loader {
  display: flex;
  justify-content: center;
  padding-top: 2rem;
}
```

Компонент Navbar

client / src / components / Navbar/ Navbar.js

```
import React, {useContext} from 'react'
import {NavLink, useHistory} from 'react-router-dom'
import {AuthContext} from "../../context/AuthContext";
import {Modal} from 'react-materialize';
import {Auth} from "../Auth/Auth";
import './Navbar.scss'


export const Navbar = () => {
  const history = useHistory()
  const {logout, isAuthenticated, userId} = useContext(AuthContext)

  const logoutHandler = (event) => {
    event.preventDefault()
    logout()
    history.push('/users')
  }

  const trigger = <li><a href="/" onClick={(event) => {event.preventDefault()}}>Log in</a></li>

  return (
    <nav className="blue darken-1">
      <div className="nav-wrapper">
        <NavLink to="/users" className="brand-logo">Банківське кредитування</NavLink>
        <ul id="nav-mobile" className="right hide-on-med-and-down">
          {
            isAuthenticated
              ? (
                <>
                  <li><NavLink to={"/company/"}>Компанії</NavLink></li>
                  <li><NavLink to={"/users/"+userId}>Cabinet </NavLink></li>
                  <li><a href="/" onClick={logoutHandler} rel="noopener noreferrer">Log out</a></li>
                </>
              )
              : <Modal
                  className="blue darken-1"
```

```
                    header="Authorization"
                    trigger={trigger}
                    fixedFooter={false}
                    options = {{preventScrolling: true}}
                  >
                    <Auth/>
                  </Modal>
              }
            </ul>
          </div>
        </nav>
    )
}
```

client / src / components / Navbar/ Navbar.js

```
nav {
  padding: 0 2rem;
}

.modal {
  width: 20%;
}

.modal {
  .modal-footer {
    display: none;
  }

  .modal-content {
    padding: 0;
    border-radius: 2px;

    h4 {
      padding-top: 16px;
      padding-left: 24px;
      margin-bottom: 0;
    }
  }
}
```

Компонент LoginBtn

client / src / components / LoginBtn.js

```
import React, {useContext, useEffect} from "react";
import {AuthContext} from "../context/AuthContext";
import {FormContext} from "../context/FormContext";
import {useMessage} from "../hooks/message.hook";
import {useHttp} from "../hooks/http.hook";


export const LoginBtn = () => {

    const {login} = useContext(AuthContext)
    const [form] = useContext(FormContext)
    const message = useMessage()
    const {loading, error, req, clearError} = useHttp(false)

    useEffect(() => {
        message(error)
        clearError()
```

```
  }, [error, message, clearError])

  const loginHandler = async () => {
    try {
      const data = await req('/api/login/', 'POST', {email: form.email, password: form.password})
      if (data.isNewUser) message('User was added')
      login(data.token, data.userId, data.balance, data.nickname)
    } catch (e) {}
  }

  return (
    <button
      className="btn blue-grey darken-4 login-btn"
      onClick={loginHandler}
      disabled={loading}
    >
      Log in
    </button>
  )
}
```

Користувацькі хуки

Хук auth.hook.js

client / src / hooks / auth.hook.js

```
import {useState, useCallback, useEffect} from 'react'

const storageName = 'userData'

export const useAuth = () => {
  const [token, setToken] = useState(null)
  const [userId, setUserId] = useState(null)
  const [ready, setReady] = useState(false)
  const [balance, setBalance] = useState(null)
  const [nickname, setNickname] = useState(null)

  const login = useCallback((jwtToken, id,balance,nickname) => {
    setToken(jwtToken)
    setUserId(id)
    setBalance(balance)
    setNickname(nickname)
    localStorage.setItem(storageName, JSON.stringify({
      userId: id,
      token: jwtToken
    }))
  }, [])

  const logout = useCallback(() => {
    setToken(null)
    setUserId(null)
    setBalance(null)
    setNickname(null)
    localStorage.removeItem(storageName)
  }, [])

  useEffect(() => {
    const data = JSON.parse(localStorage.getItem(storageName));

    if (data && data.token) {
      login(data.token, data.userId)
```

```
    }
    setReady(true)
  }, [login])

  return {login, ready, logout, token, userId}
}
```

Хук http.hook.js

client / src / hooks / http.hook.js

```
import {useState, useCallback} from 'react'

export const useHttp = () => {
  const [loading, setLoading] = useState(false)
  const [error, setError] = useState(null)

  const req = useCallback(async (url, method = 'GET', body = null, headers = {}) => {

    setLoading(true)
    try {
      if (body) body = JSON.stringify(body)
      headers['Content-Type'] = 'application/json'

      const res = await fetch(url, {method, body, headers})
      const data = await res.json()

      if (!res.ok) {
        throw new Error(data.message || 'Щось пішло не так')
      }
      setLoading(false)
      return data
    } catch (e) {
      setLoading(false)
      setError(e.message)
      throw e
    }
  }, [])

  const clearError = useCallback(() => setError(null), [])

  return {loading, req, error, clearError}
}
```

Хук message.hook.js

client / src / hooks / message.hook.js

```
import {useCallback} from 'react'

export const useMessage = () => {
  return useCallback(text => {
    if (window.M && text) {
      window.M.toast({html: text})
    }
  }, [])
}
```

Точка входу JavaScript

client / src / index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.scss';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(
 <React.StrictMode>
  <App />
 </React.StrictMode>,
 document.getElementById('root')
);

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

client / src / App.js

```
import React from 'react'
import {useRoutes} from "./routes"
import {BrowserRouter as Router} from "react-router-dom"
import {useAuth} from "./hooks/auth.hook";
import {AuthContext} from "./context/AuthContext";
import {Navbar} from "./components/Navbar/Navbar";
import 'materialize-css'
import {Loader} from "./components/Loader/Loader";


function App() {
   const {token, login, ready, logout, userId, balance, email} = useAuth()
   const isAuthenticated = !!token
   const routes = useRoutes(isAuthenticated);

   if (!ready) {
      return <Loader/>
   }

   return (
      <AuthContext.Provider value={{token, login, logout, userId, balance, email, isAuthenticated}}>
         <Router>
            <Navbar/>
            <div className="container">
               {routes}
            </div>
         </Router>
      </AuthContext.Provider>
   );
}

export default App;
```

Маршрутизація

client / src / routes.js

```
import React from "react"
import {Switch, Route, Redirect} from "react-router-dom"
```

```
import {UserPage} from "./pages/UserPage"
import {UserListPage} from "./pages/UserListPage/UserListPage"
import {CalculatorPage} from "./pages/CalculatorPage/CalculatorPage";
import {CompaniesPage} from "./pages/CompaniesPage";
import {WellcomePage} from "./pages/WellcomePage";

export const useRoutes = isAuthenticated => {
  if (isAuthenticated) {
    return (
      <Switch>
        <Route path="/company" exact>
          <CompaniesPage />
        </Route>
        <Route path="/calculator" exact>
          <CalculatorPage />
        </Route>
        <Redirect to="/calculator"/>
      </Switch>
    )
  }

  return (
    <Switch>
      <Route path="/">
        <WellcomePage />
        <Redirect to="/" />
      </Route>
    </Switch>
  )



  return (


    <Switch>
      <Route path="/users" exact>
        <UserListPage />
      </Route>

      <Route path="/calculator" exact>
        <CalculatorPage />
      </Route>
      {
        isAuthenticated && (
          <Route path="/users/:id">
            <UserPage />
          </Route>

        )
      }
      <Redirect to="/users"/>
    </Switch>
  )
}
```

Додаток Е

Back-end

Основна сторінка сервера

app.js

```
//npm run start
// для роботи з маршрутами, запуску сервера, валідації даних з фронтенду
const express = require('express')
const config = require('config')
const mongoose = require('mongoose')

const app = express()
// переводить дані req.body стрім-формату в json
app.use(express.json({extended: true}))
app.use('/api/auth', require('./routes/auth.routes'))
app.use('/api/users', require('./routes/user.routes'))
app.use('/api/company', require('./routes/company.routes'))


const PORT = config.get('PORT') || 5000
const ATLAS_URI = config.get('ATLAS_URI');

async function start() {
  try {
    await mongoose.connect(ATLAS_URI, {
      useNewUrlParser: true, useCreateIndex: true, useUnifiedTopology: true
    })
    app.listen(PORT, () => console.log(`App has been started on port ${PORT}`))
  } catch (e) {
    console.log('Server Error', e.message)
    process.exit(1)
  }
}

start();
```

Маршрути

routes / auth.routes.js

```
const {Router} = require('express')
const bcrypt = require('bcryptjs')
const config = require('config')
const jwt = require('jsonwebtoken')
const {check, validationResult} = require('express-validator')
const User = require('../models/user.model')
const router = Router()


// /api/auth/register
router.post(
  '/register',
  [
    check('email', 'Некоректний email').isEmail(),
    check('password', 'Мінімальна довжина паролю 6 символів')
      .isLength({ min: 6 })
  ],
```

```
  async (req, res) => {
    try {
      const errors = validationResult(req)

      if (!errors.isEmpty()) {
        return res.status(400).json({
          errors: errors.array(),
          message: 'Некоректні дані під час реєстрації'
        })
      }

      const {email, password} = req.body

      const candidate = await User.findOne({ email })

      if (candidate) {
        return res.status(400).json({ message: 'Такий користувач уже існує' })
      }

      const hashedPassword = await bcrypt.hash(password, 12)
      const user = new User({ email, password: hashedPassword })

      await user.save()

      res.status(201).json({ message: 'Користувач створений' })

    } catch (e) {
      res.status(500).json({ message: 'Щось пішло не так, спробуйте знову' })
    }
  })

// /api/auth/login
router.post(
  '/login',
  [
    check('email', 'Введіть коректний email').normalizeEmail().isEmail(),
    check('password', 'Введіть пароль').exists()
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req)

      if (!errors.isEmpty()) {
        return res.status(400).json({
          errors: errors.array(),
          message: 'Некоректні дані при вході в систему'
        })
      }

      const {email, password} = req.body

      const user = await User.findOne({ email })

      if (!user) {
        return res.status(400).json({ message: 'Користувач не знайдений' })
      }

      const isMatch = await bcrypt.compare(password, user.password)

      if (!isMatch) {
        return res.status(400).json({ message: 'Невірний пароль, спробуйте знову' })
      }
```

```
        const token = jwt.sign(
          { userId: user.id },
          config.get('jwtSecret'),
          { expiresIn: '1h' }
        )

        res.json({ token, userId: user.id })

      } catch (e) {
        res.status(500).json({ message: 'Щось пішло не так, спробуйте знову' })
      }
    })


module.exports = router
```

routes / company.routes.js

```
const router = require('express').Router()
const companyCtrl = require('../controllers/company.controller')
const auth = require('../middleware/auth.middleware')

router.route('/')
  .get(auth, companyCtrl.list)
  .post(companyCtrl.create)

router.route('/:id')
  .get(companyCtrl.get)
  .put(companyCtrl.update)
  .delete(companyCtrl.remove);

router.param('id', companyCtrl.load);


module.exports = router
```

routes / user.routes.js

```
const router = require('express').Router()
const userCtrl = require('../controllers/user.controller')

router.route('/')
  .get(userCtrl.list)
  .post(userCtrl.create)

router.route('/:id')
  .get(userCtrl.get)
  .put(userCtrl.update)
  .delete(userCtrl.remove);

router.param('id', userCtrl.load);


module.exports = router
```

Моделі

models / company.model.js

```
const {Schema, model} = require('mongoose')

const schema = new Schema({
    name: {
        type: String,
        required: true,
        unique: true
    },
    cash: {
        type: Number,
        required: true
    },
    STFI: {
        type: Number,
        required: true
    },
    receivables: {
        type: Number,
        required: true
    },
    SC: {
        type: Number,
        required: true
    },
    equity: {
        type: Number,
        required: true
    },
    CL: {
        type: Number,
        required: true
    },
    BST: {
        type: Number,
        required: true
    },
    GR: {
        type: Number,
        required: true
    },
    profit: {
        type: Number,
        required: true
    }
})

// 1. cash          - Cash
// 2. STFI          - Short-Term Financial Investments
// 3. receivables   - Receivables
// 4. SC            - Stocks and Costs
// 5. equity        - Equity
// 6. CL            - Current Liabilities
// 7. BST           - Balance Sheet Total
// 8. GR            - Gross Revenue
// 9. profit        - Profit

module.exports = model('company', schema)
```

models / user.model.js

```
const {Schema, model} = require('mongoose')
```

```
const schema = new Schema({
    nickname: {
      type: String,
      required: true,
      unique: true
    },
    balance: {
      type: Number,
      required: true,
      default: 0
    }
  },
  {
    versionKey: false
  }
)

module.exports = model('user', schema)
```

Контролери

controllers / auth.controller.js

```
const {validationResult} = require('express-validator');
const User = require('../models/user.model')
const jwt = require('jsonwebtoken')
const config = require('config')

const login = async (req, res) => {
  try {
    const errors = validationResult(req)
    let isNewUser = true

    if (!errors.isEmpty()) {
      return res.status(400).json({
        errors: errors.array(),
        message: errors.array().map(e => e.msg).join(' ; ')
      })
    }

    const {nickname} = req.body
    let user = await User.findOne({nickname})

    if (!user) {
      try {
        await new User({nickname}).save()
      } catch(err) {
        res.status(400).json({message: 'Error while user adding: ' + err} )
      }
      user = await User.findOne({nickname})
    } else {
      isNewUser = false
    }

    const balance = user.balance
    const token = jwt.sign(
      { userId: user.id},
      config.get('JWT_SECRET'),
      {expiresIn: '1h'}
    )
```

```
        res.json({token, user: {id: user.id, balance, nickname}, isNewUser})
    } catch(err) {
        res.status(500).json({message: 'Something went wrong, try again!_'})
    }
}

module.exports = {
    login
}
```

controllers / company.controller.js

```
const Company = require('../models/company.model')
const {Types} = require('mongoose')


const list = async (req, res, next) => {
    try {
        const list = await Company.find()
        res.json(list)
        console.log('list')
        console.log(list)
    } catch(e) {
        res.status(500).json({message: 'Something went wrong during companies fetching'})
        next(e)
    }
}

const create = async (req, res, next) => {
    try {
        const company = await new Company(req.body).save()
        res.json(company)
    } catch(e) {
        res.status(500).json({message: 'Something went wrong during company adding'})
        next(e)
    }
}

const get = (req, res)  => {
    return res.json(req.company);
}

const update = async (req, res, next) => {
    try {
        const company = Object.assign(req.company,req.body)
        const updatedCompany = await company.save()
        res.json(updatedCompany)
    } catch (e) {
        res.status(500).json({message: 'Something went wrong during company updating'})
        next(e)
    }
}

const remove = async (req, res) => {
    try {
        req.company.remove()
        res.json(req.company)
    } catch (e) {
        res.status(500).json({message: 'Something went wrong during company removing'})
    }
}
```

```
const load = async (req, res, next, id) => {
    try {
        const company = await Company.findOne({_id: Types.ObjectId(id)})
        if (!company) throw Error('Company with this id is not found')
        req.company = company
        next()
    } catch(e) {
        res.status(500).json({message: 'Something went wrong width company id'})
        next(e)
    }
}

module.exports = {list, create, get, update, remove, load}
```

controllers / user.controller.js

```
const User = require('../models/user.model')
const {Types} = require('mongoose')

const list = async (req, res, next) => {
    try {
        const list = await User.find()
        res.json(list)
    } catch(e) {
        res.status(500).json({message: 'Something went wrong during users fetching'})
        next(e)
    }
}

const create = async (req, res, next) => {
    try {
        const user = await new User(req.body).save()
        res.json(user)
    } catch(e) {
        res.status(500).json({message: 'Something went wrong during user adding'})
        next(e)
    }
}

const get = (req, res)  => {
    return res.json(req.user);
}

const update = async (req, res, next) => {
    try {
        const user = Object.assign(req.user,req.body)
        const updatedUser = await user.save()
        res.json(updatedUser)
    } catch (e) {
        res.status(500).json(e)
        next(e)
    }
}

const remove = async (req, res) => {
    try {
        req.user.remove()
        res.json(req.user)
    } catch (e) {
        res.status(500).json({message: 'Something went wrong during user removing'})
    }
}
```

```
const load = async (req, res, next, id) => {
  try {
    const user = await User.findOne({_id: Types.ObjectId(id)})
    if (!user) throw Error('User with this id is not found')
    req.user = user
    next()
  } catch(e) {
    res.status(500).json({message: 'Something went wrong width user id'})
    next(e)
  }
}

module.exports = {list, create, get, update, remove, load}
```

Проміжне програмне забезпечення

middleware / auth.middleware.js

```
const jwt = require('jsonwebtoken')
const config = require('config')

module.exports = (req, res, next) => {

  if (req.method === 'OPTIONS') {
    return next()
  }

  try {
    const token = req.headers.authorization.split(' ')[1]

    if (!token) {
      return res.status(401).json({message: 'Цей користувач не авторизований'})
    }

    const decoded = jwt.verify(token, config.get('JWT_SECRET'))
    req.user = decoded
    next()
  } catch (e) {
    res.status(401).json({message: 'Цей користувач не авторизований'})
  }
}
```