КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Кафедра інформаційних технологій

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

Розробка підсистеми оцінки пошкоджень будівель на основі нечіткої логіки

Шульга Денис Мирославович

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «**Розробка підсистеми оцінки пошкоджень будівель на основі нечіткої логіки**»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Шульга Денис Мирославович
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21
Керівник Горда О. В.
(прізвище та ініціали)
к.т.н., доцент
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Баліна О.І.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп'ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___" _____2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА

ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ **БАКАЛАВР**

	Шульзі Денису Мирославовичу

1. Тема роботи - **Розробка підсистеми оцінки пошкоджень будівель на основі нечіткої логіки**

затверджена наказом ректора КНУБА № 235/23/25 від «14» лютого 2025 року

2. Керівник роботи	Горда Олена Володимирівна, к.т.н. доцент

3. Строк подання Здобувачем роботи до захисту травень 2025 р.

4. Зміст пояснювальної записки за розділами:

Р.1 ТЕОРЕТИЧНІ ТА АНАЛІТИЧНІ ОСНОВИ ОЦІНКИ ТЕХНІЧНОГО СТАНУ БУДІВЕЛЬ

Р.2 СПЕЦИФІКАЦІЯ ВИМОГ, МЕТОДИ І МОДЕЛІ

Р.3 РЕАЛІЗАЦІЯ ТА ПРАКТИЧНА АПРОБАЦІЯ ПРОГРАМНОЇ ПІДСИСТЕМИ

Р.4 ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

5. Графічний матеріал за розділами:

Робота викладена на 66 аркушах, містить 3 додатки, 5 таблиць, 17 рисунків, список використаної літератури із 20 найменувань.

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	02.02.2025
Розділ 2	03.04.2025
Розділ 3	11.05.2025
Розділ 4	17.05.2025
Остаточне оформлення роботи	23.05.2025
Направлення роботи для перевірки на плагіат	24.05.2025
Попередній захист роботи на випусковій кафедрі	26.05.2025
Направлення роботи на рецензування	28.05.2025

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1	Горда О.В., доц.каф.ІТ	03.02.2025	
Розділ 2	Рябчун Ю.В., доц.каф.ІТ	04.04.2025	
Розділ 3	Мацієвський О. О., асистент.каф.ІТ	12.05.2025	
Розділ 4	Рябчун Ю.В., доц.каф.ІТ	18.05.205	

8. Дата видачі завдання листопад 2025 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Горда О. В.
	(підпис)		(прізвище та ініціали)
Здобувач			Шульга Д.М.
	(підпис)		(прізвище та ініціали)

<table>
<tr><td>РЕЗЮМЕ (SUMMARY) до атестаційної випускної роботи Здобувача:</td><td colspan="3">Шульга Денис Мирославович

Denis Shulga</td></tr>
<tr><td>ЗВО</td><td colspan="3">Київський національний університет будівництва і архітектури</td></tr>
<tr><td>Тема (українською та англійською)</td><td colspan="3">Розробка підсистеми оцінки пошкоджень будівель на основі нечіткої логіки
Development of a building damage assessment subsystem based on fuzzy logic</td></tr>
<tr><td>Освітній ступінь</td><td colspan="3">Бакалавр</td></tr>
<tr><td>Факультет</td><td colspan="3">Автоматизації і інформаційних технологій</td></tr>
<tr><td>Випускаюча кафедра</td><td colspan="3">Інформаційних технологій</td></tr>
<tr><td>Спеціальність</td><td colspan="3">122 «Комп’ютерні науки»</td></tr>
<tr><td>Освітня програма</td><td colspan="3">Інформаційні управляючі системи та технології</td></tr>
<tr><td>Керівник</td><td colspan="3">Горда Олена Володимирівна</td></tr>
<tr><td rowspan="2">Обсяг роботи:</td><td>пояснювальна записка, стор.</td><td>розділів</td><td>креслень формату А</td></tr>
<tr><td>66</td><td>4</td><td>0</td></tr>
<tr><td>Ключові слова:

Keywords:</td><td colspan="3">нечітка логіка, оцінка пошкоджень, Dash, Python, Damage Index, технічний стан будівель, експертна система, інтерфейс користувача, SQLite, фазифікація.
fuzzy logic, damage assessment, Dash, Python, Damage Index, technical condition of buildings, expert system, user interface, SQLite, fuzzification.</td></tr>
</table>

У роботі розроблено систему виявлення жестів рук у відеопотоці з використанням нейронної мережі MediaPipe. Реалізовано програмний продукт, що автоматизує процес розпізнавання жестів у реальному часі для застосування в безконтактних інтерфейсах.

Здобувач: _____ /Денис ШУЛЬГА/

Керівник: _____/ Олена ГОРДА/

“___” _____ 202_р.

АНОТАЦІЯ

Шульга Денис Мирославович Розробка підсистеми оцінки пошкоджень будівель на основі нечіткої логіки.

Атестаційна випускна робота бакалавра за спеціальністю 122 «Комп'ютерні науки», освітня програма «Інформаційні управляючі системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Робота присвячена проєктуванню та реалізації веборієнтованої підсистеми для попереднього оцінювання технічного стану будівельних конструкцій за допомогою методів нечіткої логіки. У якості вхідних даних система приймає п'ять параметрів: ступінь пошкодження тріщинами, корозією, деформацією, розхитуванням з'єднань та загальний показник якості матеріалу. Значення нормалізуються та обробляються через механізм нечіткого виведення типу Мамдані з подальшою дефазифікацією, що дозволяє обчислити Damage Index у межах [0–1] та класифікувати пошкодження як Minor, Moderate або Significant.

Функціональність реалізовано у вигляді Dash-додатку на мові Python із візуалізацією функцій приналежності, інтерактивним інтерфейсом користувача та системою експорту результатів у форматах CSV та HTML. Також реалізовано збереження результатів у базу даних SQLite.

Результати дослідження свідчать про високу ефективність застосування нечіткої логіки для задач експертної оцінки стану об'єктів, зокрема в умовах неповної інформації або візуального огляду. Підсистема рекомендована до використання у технічному нагляді, інспекційній практиці та в системах цифрової інвентаризації об'єктів.

Робота викладена на 66 аркушах, містить 4 додатки, 5 таблиці, 17 рисунків, список використаної літератури з 20 найменувань.
Ключові слова**:** нечітка логіка, оцінка пошкоджень, Dash, Python, Damage Index, технічний стан будівель, експертна система, інтерфейс користувача, SQLite, фазифікація.

SUMMARY

Shulga D. M. Development of a subsystem for assessing damage to buildings based on fuzzy logic.

Bachelor's thesis in the field of study 122 “Computer Science,” educational program “Information Management Systems and Technologies.” – Kyiv National University of Construction and Architecture. – Kyiv, 2025.

The work is devoted to the design and implementation of a web-oriented subsystem for preliminary assessment of the technical condition of building structures using fuzzy logic methods. The system accepts five parameters as input data: the degree of damage by cracks, corrosion, deformation, loosening of joints, and the overall material quality index. The values are normalized and processed through a Mamdani-type fuzzy inference mechanism with subsequent defuzzification, which allows calculating the Damage Index within the range [0–1] and classifying damage as Minor, Moderate, or Significant.

The functionality is implemented as a Dash application in Python with visualization of membership functions, an interactive user interface, and a system for exporting results in CSV and HTML formats. Saving results to an SQLite database is also implemented.

The results of the study indicate the high efficiency of applying fuzzy logic to tasks of expert assessment of the condition of objects, in particular in conditions of incomplete information or visual inspection. The subsystem is recommended for use in technical supervision, inspection practice, and digital inventory systems for objects.

The work is presented on 66 pages and contains 4 appendices, 5 tables, 17 figures, and a list of 20 references.

Keywords: fuzzy logic, damage assessment, Dash, Python, Damage Index, technical condition of buildings, expert system, user interface, SQLite, fuzzification.

ЗМІСТ

ВСТУП

Сучасний стан розвитку інженерної галузі вимагає впровадження нових інформаційних технологій, які здатні не лише автоматизувати рутинні процеси, але й підтримувати прийняття складних технічних рішень в умовах неповної або нечіткої інформації. Особливо актуально це питання постає у сфері технічного діагностування, оцінювання технічного стану будівель та споруд, де велика кількість факторів має суб'єктивний або експертний характер, а методики аналізу залишаються застарілими або громіздкими. Традиційні методи оцінювання пошкоджень часто ґрунтуються на візуальному огляді, заповненні табличних форм, ручних розрахунках та прийнятті рішень на підставі досвіду фахівця, що призводить до варіативності у висновках, уповільнення темпів роботи та складності в стандартизації процедур.

У період після широкомасштабних руйнувань, спричинених як збройними конфліктами, так і природними катастрофами, відновлення будівельної інфраструктури набуває ключового значення. При цьому надзвичайно важливо мати у своєму розпорядженні надійні, гнучкі та адаптивні інструменти для первинної оцінки пошкоджень конструкцій, особливо у випадках, коли проведення детального інженерного обстеження потребує часу або ресурсів. У таких умовах на перший план виходить необхідність створення підсистем, які б дозволяли оперативно, із мінімальною участю людського фактору, визначити ступінь пошкодження конструкції, її експлуатаційну придатність та пріоритетність подальших дій — від локального ремонту до повної реконструкції.

У цьому контексті особливу увагу привертають системи, що базуються на принципах нечіткої логіки. Нечітка логіка — це розділ математичної логіки, який дозволяє працювати з лінгвістичними змінними, описаними в термінах «високий», «середній», «низький» тощо, а не точними числовими значеннями. Саме така форма представлення інформації максимально наближена до того, як інженер або технік інтерпретує стан конструкції при візуальному огляді. Це дає змогу формалізувати процеси експертної оцінки, не вдаючись до складної статистики або ймовірнісного аналізу.

У цій дипломній роботі поставлено завдання створити повноцінну програмну підсистему, що дозволяє реалізувати автоматизовану оцінку пошкоджень будівель на основі нечіткої логіки, із можливістю подальшої візуалізації результатів, формування висновків та збереження у базу даних. Підсистема має функціонувати як веб-застосунок із зручним інтерфейсом, придатним для використання як у мобільному, так і в настільному середовищі. Вибір платформи Dash для реалізації обумовлений її гнучкістю, широкими можливостями для інтерактивної побудови графіків, простотою розгортання та сумісністю з інструментами Python — мовою, яка активно використовується в інженерних та науково-аналітичних розробках.

Особливу увагу приділено модульному підходу до побудови системи: ядро логіки нечіткого виведення реалізоване у вигляді окремого модуля, що може функціонувати незалежно від інтерфейсу, а це відкриває перспективи його подальшого використання у мобільних додатках, REST-сервісах або вбудованих системах. Важливо, що вся база знань, тобто набір правил нечіткого виведення, також формалізована і зберігається у програмі у вигляді словникових структур, що дозволяє за потреби змінювати логіку без переписування коду.

Об’єктом дослідження у роботі виступає процес експертної оцінки технічного стану будівельних конструкцій. Предмет дослідження — методи автоматизованої інтерпретації пошкоджень із використанням інструментів нечіткої логіки в програмному середовищі. Актуальність обумовлюється практичною потребою у прискореній оцінці великої кількості об’єктів, зокрема в умовах реконструкції житлового фонду та громадської інфраструктури в посткризовий період.

Метою роботи є розробка працездатної, інтуїтивно зрозумілої та адаптивної підсистеми оцінки пошкоджень будівель, що дозволяє автоматично визначати ступінь пошкодження на підставі набору вхідних параметрів, заданих у вигляді відсоткових оцінок, та класифікувати результати за ступенем критичності. Досягнення цієї мети передбачає розв’язання низки взаємопов’язаних завдань: формалізацію вхідних ознак у вигляді нечітких множин; побудову бази правил типу Mamdani; реалізацію інтерфейсу користувача з можливістю введення даних,

перегляду графіків, таблиць і текстових висновків; забезпечення збереження результатів до локальної бази даних.

Структурно робота поділена на чотири основні розділи. У першому розділі проведено аналітичний огляд існуючих методик оцінювання пошкоджень, а також розкрито теоретичні основи нечіткої логіки. Другий розділ присвячено специфікації вимог до системи, формалізації вхідних даних і побудові логічної моделі нечіткого виведення. У третьому розділі детально описано процес реалізації програмної частини, архітектуру модулів, логіку взаємодії, побудову інтерфейсу та результати тестування. Четвертий розділ має прикладний характер і містить оцінку ергономіки інтерфейсу, техніко-економічне обґрунтування, порівняння з аналогічними рішеннями, а також обґрунтування потенціалу масштабування і розвитку системи.

Розділ 1. ТЕОРЕТИЧНІ ТА АНАЛІТИЧНІ ОСНОВИ ОЦІНКИ ТЕХНІЧНОГО СТАНУ БУДІВЕЛЬ

1.1. Технічна еволюція підходів до розпізнавання жестів

Проблема оцінки технічного стану будівель була й залишається одним із найважливіших завдань у сфері будівництва, архітектури, експлуатації та реконструкції об’єктів нерухомості. Її актуальність зумовлена тим, що технічний стан будівель безпосередньо впливає на безпеку життя та здоров’я громадян, надійність інфраструктури, ефективність експлуатації й обґрунтованість інвестицій у нерухомість [1].

З плином часу, під впливом зовнішніх чинників, таких як атмосферні опади, зміни температури, вібрації, сейсмічна активність, промислові навантаження та інтенсивна експлуатація, стан будівельних конструкцій невпинно погіршується. У багатьох випадках це погіршення носить поступовий характер, що ускладнює його своєчасне виявлення[2]. На практиці візуальна оцінка або навіть регулярні інспекції не завжди дозволяють вчасно зафіксувати зміни, які з часом можуть призвести до аварійних ситуацій. Тому питання комплексної діагностики технічного стану є не лише технічним, але й соціально значущим [3].

Технічний стан будівлі за своєю суттю — це узагальнене поняття, яке відображає рівень збереження функціональних і конструктивних характеристик будівлі або споруди у порівнянні з початковим (проектним) рівнем[4]. Це поняття включає в себе цілу низку аспектів: геометричну стабільність елементів, несучу здатність конструкцій, ступінь зносу матеріалів, наявність або відсутність пошкоджень (тріщин, деформацій, корозії), цілісність з’єднань, герметичність огороджувальних конструкцій, ефективність тепло- та звукоізоляції тощо. Іншими словами, технічний стан — це багатовимірна категорія, яка не може бути охарактеризована одним параметром чи простою шкалою [5].

Серед ключових понять, тісно пов’язаних із технічним станом будівель, варто окремо розглянути такі терміни, як **“**експлуатаційна придатність**”** та

“критичність пошкоджень”. Експлуатаційна придатність — це здатність будівлі або її окремих конструктивних елементів виконувати функції, для яких вони були призначені, в умовах реальної експлуатації [6]. Це поняття передбачає, що будівля не лише фізично існує, але й функціонує в межах нормативних значень: не деформується надмірно, не спричиняє небезпеки, не створює перешкод для повсякденного використання. Критичність пошкоджень, у свою чергу, відображає ту межу, за якої подальше використання конструктивного елемента або будівлі в цілому становить загрозу [7]. Часто критичність не залежить лише від розміру тріщини або глибини корозії, але й від розташування дефекту, типу конструкції, характеру навантажень тощо, зображено на рисунку 1.1.

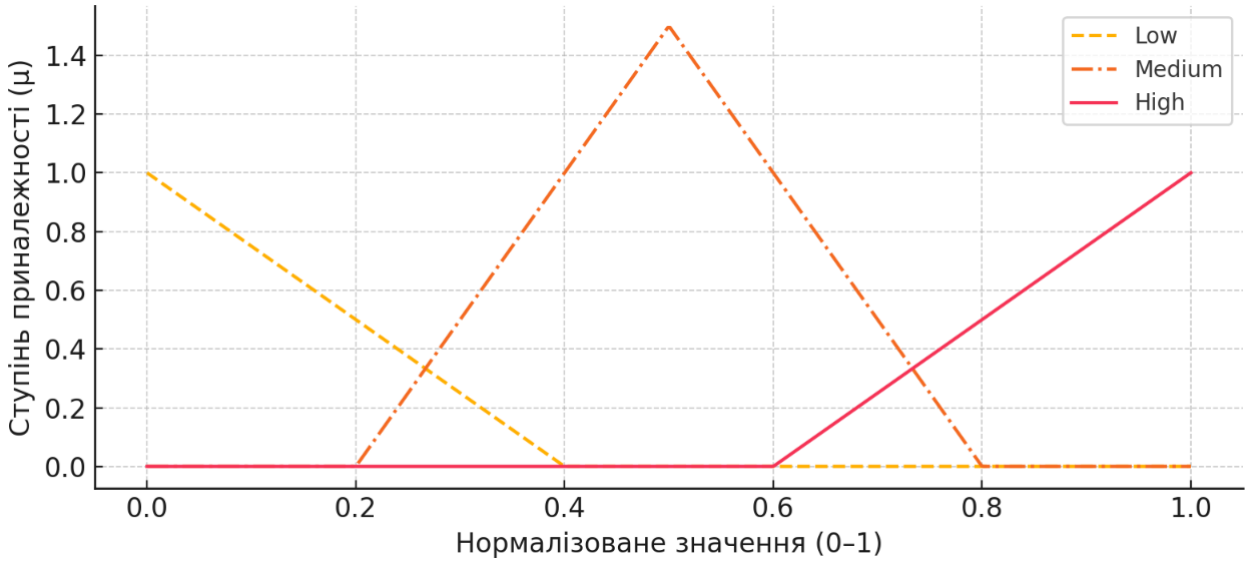


Рисунок 1.1. Типова будівельна конструкція з пошкодженнями

Сучасна практика експлуатації будівель показує, що значна частина об'єктів, зокрема в пострадянських країнах, функціонує вже давно за межами розрахункового терміну експлуатації. Це особливо актуально для житлових будинків масової забудови 1960–1980-х років, які були зведені із розрахунком на 40–50 років служби, але нині продовжують використовуватися. В умовах недостатнього фінансування капітального ремонту, відсутності системного моніторингу та недосконалості регуляторної бази, контроль за технічним станом часто здійснюється поверхово або взагалі ігнорується. Такі обставини створюють

передумови для техногенних катастроф — обвалів, просідань, аварій інженерних систем, загроз життю мешканців.

Таким чином, на перший план виходить завдання створення інструментів, які б дозволяли об’єктивно, з мінімальними витратами часу та ресурсів, але з достатнім ступенем точності оцінювати технічний стан будівель. Саме в цьому контексті виникає потреба в розробці інтелектуальних підсистем діагностики, зокрема — на основі методів нечіткої логіки, що здатні працювати з неповними, неточними чи суб’єктивними даними, які часто трапляються у сфері будівельно-технічної експертизи.

1.1.1.. Технічна еволюція підходів до розпізнавання жестів

Пошкодження будівельних конструкцій — це складний і багатофакторний процес, який може відбуватись поступово або миттєво, внаслідок короткочасного впливу значного навантаження [8]. Залежно від природи, механізму виникнення, розташування та характеру впливу, пошкодження умовно поділяють на декілька ключових типів, кожен з яких має свої особливості прояву, діагностики та впливу на загальний технічний стан об’єкта.

Одним із найпоширеніших типів ушкоджень є механічні тріщини (crack damage) [9]. Тріщини виникають унаслідок перевищення розрахункових напружень, усадки матеріалу, температурних коливань або через інші технологічні та експлуатаційні причини. Візуально тріщини можуть виглядати як вертикальні, горизонтальні, сітчасті, косі, поверхневі або наскрізні. Хоча не всі тріщини становлять небезпеку, саме за ними часто оцінюється загальний стан конструкцій, адже вони свідчать про зміну напруженого стану. Особливу увагу слід приділяти тріщинам у несучих елементах (стінах, балках, плитах перекриття), оскільки вони можуть стати передвісниками втрати стійкості.

Другим важливим типом є корозійні ушкодження **(**corrosion damage**).** Корозія здебільшого стосується залізобетонних і металевих елементів і виникає внаслідок впливу вологи, агресивних середовищ, електрохімічних процесів [10]. Зовні корозійні ушкодження можуть проявлятися у вигляді іржі, спучення бетону, розшарувань, появи солей на поверхні, оголення та знищення арматури. Корозія істотно знижує несучу здатність елементів, адже зменшується ефективний переріз арматури, а хімічні реакції можуть спричинити розриви бетону. Важливо зазначити, що корозійні ушкодження здатні розвиватися приховано, особливо у місцях з недостатньою вентиляцією або деформаційною герметичністю.

Не менш поширеними є деформації (deform damage) — зміни геометричних характеристик елементів без їх очевидного руйнування. Вони проявляються у вигляді прогинів, викривлень, просідань, перекосів, що призводить до зміщення геометрії конструкції [11]. Деформації можуть бути як еластичними (що зникають після зняття навантаження), так і пластичними (незворотні зміни). Часто вони виникають унаслідок тривалого перевантаження, нерівномірної осадки ґрунту, усадки бетону або старіння матеріалів. Деформації здатні погіршувати архітектурні та функціональні властивості об’єкта: викликати перекоси дверей і вікон, порушення гідроізоляції, розгерметизацію швів.

Ще один важливий тип пошкоджень — розхитування та розшарування з’єднань (joint looseness) [12]. У будівельних конструкціях з’єднання відіграють критичну роль, оскільки саме через них передаються навантаження між елементами. Розхитування з’єднань часто є наслідком повторюваних динамічних впливів, недостатньо жорсткого монтажу, деформацій або втрати фіксації болтів, зварних швів, анкерів. Такий тип пошкоджень призводить до того, що конструкція втрачає цілісність, елементи починають працювати не як єдине ціле, а незалежно, що призводить до підвищених локальних напружень і подальших руйнувань.

Окремої уваги заслуговує зниження якості матеріалів (material quality), яке зазвичай виникає внаслідок природного старіння, замокання, заморожування, впливу ультрафіолету, дії агресивних середовищ. Це зниження проявляється у втраті міцності, порушенні структури матеріалу, виникненні мікротріщин, зменшенні щільності тощо [13]. Воно часто є непрямим показником, але суттєво впливає на здатність конструкції протистояти навантаженням.

Усі ці типи пошкоджень можуть проявлятись окремо або в комбінаціях. Саме тому ефективна система оцінки технічного стану має враховувати кожен із них, з можливістю взаємного впливу. У реальних умовах експлуатації жодне ушкодження не діє в ізоляції: тріщини сприяють проникненню вологи, що запускає корозійні процеси; деформації викликають розхитування з’єднань, а зниження якості матеріалів ускладнює оцінку реальної несучої здатності.

1.2. Постановка та аналіз проблеми

Оцінка технічного стану будівель є не лише суто інженерним завданням, але й критичним аспектом забезпечення безпеки, збереження матеріальних цінностей і запобігання соціальним втратам. Несвоєчасне або некоректне виявлення пошкоджень може мати катастрофічні наслідки як у короткостроковій, так і в довгостроковій перспективі.

Першим і найважливішим наслідком є ризик обвалу конструкцій. У випадку, коли несучі елементи втрачають свою міцність, стійкість або цілісність, будівля перестає бути безпечною для перебування людей. Історія знає чимало прикладів, коли ігнорування ранніх ознак тріщин, деформацій або розшарувань призводило до обвалу міжповерхових перекриттів, падіння фасадних елементів, руйнування балконів, підпірних стін та інших фрагментів. Особливо небезпечні ситуації виникають у громадських будівлях, навчальних закладах, лікарнях, де перебуває велика кількість людей.

Другим суттєвим наслідком є економічні збитки, які виникають унаслідок передчасного зносу конструкцій або необхідності виконання термінових (і часто дорожчих) ремонтних робіт. Замість планового капітального ремонту власник будівлі змушений проводити аварійно-відновлювальні заходи, які часто потребують більше ресурсів, часу і призводять до порушення нормального функціонування об’єкта. Якщо йдеться про об'єкти бізнесу, підприємства або складські приміщення, простої можуть спричинити втрату доходу, порушення логістики чи договорів постачання.

Також слід відзначити соціальні наслідки, що пов’язані із зниженням довіри до органів контролю, служб ЖКГ або забудовників. Якщо будівля, яка нещодавно пройшла перевірку або була здана в експлуатацію, демонструє ознаки руйнування, це викликає обурення суспільства, погіршує репутацію інституцій, породжує сумніви щодо прозорості процедур.

Не менш важливою є довгострокова деградація міського середовища. Якщо на пошкоджені будинки не звертають увагу роками, вони поступово перетворюються на джерело небезпеки, руйнують архітектурний вигляд міста, стимулюють депопуляцію районів, де розміщені. Часто такі зони перетворюються на маргіналізовані території з високим рівнем злочинності.

Таким чином, адекватна, своєчасна і технологічно підтримана система оцінки технічного стану будівель — це не лише про інженерію. Це — про безпеку життя, збереження активів, стабільність міст і довіру громадян до інфраструктури. Тому створення нових підходів, зокрема таких, що базуються на методах нечіткої логіки, дозволяє не лише автоматизувати процеси, але й зробити їх стійкими до суб’єктивних помилок, надавши експертам гнучкий і зрозумілий інструмент для ухвалення рішень.

1.2. Огляд існуючих методів оцінки технічного стану будівель

Питання визначення технічного стану будівель не є новим у будівельній галузі — воно існувало з моменту виникнення першого постійного житла і набуло системного характеру з розвитком інженерії, нормативного регулювання,

урбанізації та стандартизації будівництва. Протягом десятиліть у різних країнах напрацьовувалися підходи до інспекції будівель, оцінки їхнього стану, прогнозування довговічності й розрахунку залишкового ресурсу. У цьому контексті сформувалися три основні напрями, які сьогодні мають практичне застосування: класичні методи інспекційного контролю, параметричні розрахункові методи, а також сучасні підходи на основі штучного інтелекту та інтелектуального аналізу даних.

1.2.1. Класичні методи інспекційного контролю

На пострадянському просторі, а також у багатьох країнах Європи та Азії, основою для оцінки технічного стану будівель слугують традиційні методи інспекційного контролю. Їх сутність полягає у візуальному огляді конструкцій кваліфікованим фахівцем або комісією з наступним зняттям показників за допомогою інструментальних засобів, вимірювальних приладів та шаблонів [15]. За цими даними складається технічний паспорт об’єкта, в якому фіксується стан кожного конструктивного елементу, його пошкодження, ступінь зносу та потреба в ремонті чи заміні.

До найбільш поширених методів належать:

- огляд тріщин із фіксацією їх розміру, довжини, напрямку розповсюдження;

перевірка наявності корозії з візуальною оцінкою ступеня її розвитку;

- інструментальні заміри прогинів, перекосів, деформацій за допомогою рулеток, нівелірів, геодезичних приладів;
- застосування пристроїв для вимірювання міцності бетону на стиснення (молоток Шмідта, ультразвукові тестери).

Основною перевагою класичних методів є їх простота, доступність та відповідність нормативній базі. В Україні широке застосування мають вимоги ДСТУ та ДБН, зокрема:

- ДСТУ-Н Б В.1.2-18:2016 "Настанова з обстеження будівельних конструкцій будівель і споруд";
- ДБН В.1.2-14:2018 "Загальні принципи забезпечення надійності та конструктивної безпеки будівель та споруд";
- ДСТУ Б В.2.6-156:2010, що встановлює методику оцінки бетонних конструкцій.

Разом із тим, класичні методи мають низку суттєвих недоліків. Вони значною мірою залежать від суб’єктивності експерта, часто не дають кількісної оцінки ризиків і не враховують можливість комбінованої дії кількох ушкоджень одночасно. Крім того, у великих об'єктах обстеження вимагає багато часу, людських ресурсів, а іноді й спеціальної техніки для доступу до важкодоступних місць.

Таким чином, хоча класичний підхід залишається основою нормативного регулювання в більшості країн, він потребує доповнення сучасними аналітичними інструментами для забезпечення оперативності, точності та об'єктивності оцінок.

1.2.2. Оцінка за допомогою параметричних індексів

З метою зменшення суб’єктивності інспекцій і для підвищення об’єктивності прийняття рішень у практиці технічного обстеження будівель почали застосовуватись параметричні методи оцінки стану, в основі яких лежать розрахунки за формалізованими формулами [16].

Серед найбільш поширених параметрів у таких методах виділяють: коефіцієнти зносу, частотні коефіцієнти пошкодження, індекси надійності, а також індекс пошкодження (Damage Index). Ці коефіцієнти дозволяють на основі значень вимірюваних параметрів (наприклад, відсоток тріщин, зміна розмірів елементів, втрата міцності) розрахувати єдину числову оцінку, що відображає загальний технічний стан будівлі або її елементів.

Різні методики використовують різні шкали оцінювання. Наприклад, коефіцієнт технічного стану K_i може бути обчислений за формулою:

$$K_i = 1 - \frac{\sum_{i=1}^{n} w_i \cdot d_i}{\sum_{i=1}^{n} w_i} \quad (1)$$

де:

w_i - ваговий коефіцієнт важливості і-го дефекту;

d_i - ступінь прояву дефекту і-го типу (в балах або %);

n - кількість врахованих дефектів.

Таким чином, якщо відомі ваги і значення параметрів, можна об’єктивно обчислити загальну величину, що підлягає порівнянню з нормативними або граничними значеннями.

На практиці подібні методи застосовуються в діяльності ОСББ, технічних відділів експлуатації житла, проектних та експертних організацій, особливо коли мова йде про обґрунтування необхідності фінансування капітального ремонту. Наприклад, організації, які обслуговують багатоквартирні будинки, ведуть облікові картки технічного стану, де фіксуються показники пошкоджень та розраховується індекс стану на кожному етапі експлуатації.

Перевага параметричних методів полягає у їх кількісному характері та можливості автоматизації розрахунків. Недолік — неадаптивність до ситуацій, коли деякі параметри неможливо точно виміряти або значення є суб’єктивними (наприклад, “легка іржа” чи “значна тріщина” можуть бути інтерпретовані по-різному). В таких умовах виникає потреба у більш гнучких методах аналізу.

1.2.3. Методи, що використовують штучний інтелект

Зі стрімким розвитком інформаційних технологій у другій половині XX та на початку XXI століття в інженерній практиці почали впроваджуватись методи штучного інтелекту (ШІ). Це новий клас підходів, здатних виявляти закономірності у складних, багатовимірних і нечітких даних, які традиційні методи не в змозі обробити ефективно. У сфері технічної діагностики будівель ШІ демонструє значний потенціал.

До найбільш поширених підходів у цьому напрямі належать:

- нейронні мережі, які навчаються на історичних даних і здатні прогнозувати стан конструкції за певними вхідними параметрами;
- експертні системи, що базуються на наборах правил типу «якщо… тоді…» і імітують міркування досвідченого інженера;
- генетичні алгоритми, які дозволяють автоматично оптимізувати структуру оцінювання, підбираючи найкращі комбінації параметрів та коефіцієнтів.

Особливе місце займає нечітка логіка (fuzzy logic), яка є складовою м'яких обчислень і дозволяє моделювати процеси прийняття рішень у ситуаціях з невизначеністю. У дослідженнях, проведених в Японії, США, Німеччині та Китаї, нечітка логіка неодноразово показала ефективність у задачах оцінки технічного стану, коли параметри важко точно виміряти або коли виникає потреба поєднати декілька суб'єктивних оцінок експертів.

Зокрема, у публікації "Fuzzy logic-based method for assessing structural damage in concrete bridges" (Journal of Civil Engineering), група дослідників запропонувала методику побудови системи нечіткого виведення, яка враховує шість ознак пошкодження і визначає рівень ризику. Аналогічні підходи були реалізовані в Італії, Кореї та Канаді — у вигляді мобільних додатків, веб-сервісів або вбудованих у БІМ-системи (Building Information Modeling).

Перевага методів ШІ полягає в здатності адаптуватися до нових даних, працювати з неповною інформацією, виявляти приховані закономірності. Основний недолік — складність у поясненні рішень та необхідність якісних навчальних даних. Тим не менш, у галузі будівельної експертизи нечіткі системи мають один із найвищих потенціалів через свою здатність працювати з лінгвістичними змінними ("високий рівень тріщин", "низька якість матеріалу" тощо).

Підсумовуючи вищезазначене, можна зробити висновок, що кожен із методів оцінки технічного стану будівель має свої переваги та обмеження. Класичні методи є найбільш регламентованими, але вони залежать від людського фактора і не

завжди забезпечують точність. Параметричні методи дозволяють частково формалізувати оцінювання, проте їхній застосунок обмежується випадками, де можна однозначно встановити всі вхідні параметри. У свою чергу, інтелектуальні методи, зокрема нечітка логіка, забезпечують гнучкість, універсальність та адаптивність, але потребують певної технічної культури у користувача та правильно побудованої бази правил.

Нечітка логіка найбільш доречна там, де дані є неповними, неточними, нечіткими або суб’єктивними — саме такою часто є інформація при первинному огляді будівель без детального інструментального дослідження. У таких умовах класичні методи часто безсилі або дають суперечливі результати, тоді як нечіткі системи дозволяють зробити обґрунтований висновок, навіть якщо частина даних є наближеною або описаною словами, а не точними числами.

З огляду на це, розробка підсистеми оцінки технічного стану будівель із використанням методів нечіткої логіки виглядає цілком обґрунтованою та актуальною як для практичного впровадження, так і для подальших досліджень.

1.3. Теоретичні основи нечіткої логіки та нечіткого виведення

У процесі дослідження та моделювання технічного стану будівель усе частіше доводиться стикатися з ситуаціями, коли неможливо або надто складно надати об’єктивну, точну й чисельно формалізовану оцінку параметрів об’єкта. Особливо це стосується характеристик, що описуються не стільки точними цифровими значеннями, скільки якісними [16], суб’єктивними або наближеними категоріями — такими як "висока якість матеріалу", "помітна тріщина", "значна корозія" тощо. У таких випадках класичні методи математичного аналізу, основані на жорстких логічних схемах, виявляються недостатньо гнучкими, а подекуди й непридатними для адекватного відображення реальності.

Саме в таких умовах на передній план виходить концепція нечіткої логіки (fuzzy logic) — наукового напряму, який дозволяє працювати з невизначеністю, лінгвістичними змінними та наближеними судженнями. Запропонована у 1965 році американським вченим іранського походження Лотфі Заде (Lotfi A. Zadeh), нечітка

логіка стала революційним підходом у теорії множин, який запропонував альтернативу класичному дихотомічному мисленню: замість «так» і «ні» — «в певній мірі так» і «в певній мірі ні». Такий підхід виявився надзвичайно природним для опису міркувань людини, яка зазвичай не мислить у категоріях 0 або 1, а скоріше оперує поняттями «більше», «менше», «майже», «дещо», «значно» тощо.

У цьому підрозділі буде здійснено розгорнутий огляд базових понять нечіткої логіки, основних структур нечіткого виведення та особливостей їх застосування саме у сфері оцінювання технічного стану будівель. Окрема увага приділена механізмам формалізації експертних суджень, структуруванню лінгвістичних змінних та методам інтерпретації результатів у вигляді кількісного висновку.

1.3.1. Основні поняття нечіткої логіки

Нечітка логіка, як галузь знання, базується на загальній ідеї розмитої приналежності, що дозволяє будь-якому елементу входити до множини не лише повністю, а й частково. На відміну від класичної множини, у якій елемент або належить (1), або не належить (0), нечітка множина дозволяє ступені належності в інтервалі від 0 до 1.

Наприклад, у класичному випадку ми могли б сказати: «бетон міцний», тобто він належить до множини "міцних матеріалів". Однак у реальності це твердження може бути тільки наближеним. Можливо, бетон вже частково втратив свої властивості внаслідок старіння або корозії, але ще не настільки, щоб вважати його повністю непридатним. У цьому випадку твердження "бетон є міцним" може бути істинним лише, скажімо, на 0.7. Тобто його ступінь приналежності до множини «міцний матеріал» — 0.7.

Цей ступінь задається функцією приналежності (Membership Function, MF). Функція приналежності — це графічне або аналітичне відображення того, як змінюється міра належності до нечіткої категорії залежно від значення змінної. У будівельній практиці це можуть бути такі змінні, як товщина тріщини, вологість матеріалу, рівень прогину балки тощо [17].

Функції приналежності можуть набувати різної форми (таблиця 1.1.):

- трикутна — найпростіша, симетрична, визначається трьома точками (ліва межа, пік, права межа);
- трапецієвидна — має плато, де значення приналежності дорівнює 1 (ідеально для позначення «нормальних» зон);
- гаусова (нормальна) — задається експоненційною кривою, м'яко зростає і спадає, зручно для опису біомеханічних або фізичних процесів.

Таблиця 1.1

Типи функцій приналежності

Тип функції приналежності	Графічна форма	Сфера застосування	Переваги	Недоліки
Трикутна	Пікообразна	Прості моделі, лінійні переходи	Легкість реалізації	Грубі краї переходу
Трапецієвидна	З плато	Стабільні категорії, нечутливі до дрібних коливань	Гнучкість і простота	Нечітке визначення центру
Гаусова	Крива дзвону	Плавні розподіли, біомеханіка, нечіткі статистичні дані	Висока плавність переходів	Складність обчислень

Поняття лінгвістичної змінної є одним із центральних у нечіткій логіці [18]. Це змінна, значення якої виражається словами природної мови. Наприклад, змінна «стан матеріалу» може мати терми: «низький», «середній», «високий». Для кожного з цих термів будується окрема функція приналежності.

На відміну від цифрових змінних, лінгвістичні змінні зручні для експертів — вони дозволяють формулювати правила у звичній мовній формі. Це спрощує процес створення моделей та інтерфейс взаємодії із користувачем системи.

Ще одним важливим поняттям є дефазифікація — перетворення розмитого результату назад у чітке значення. Вона потрібна для того, щоб вивести кінцевий

числовий результат: наприклад, якщо система визначила, що об'єкт має 0.8 ступеня належності до «значного пошкодження» і 0.3 — до «помірного», то ми повинні обчислити одну числову оцінку, наприклад, 0.74, яка буде відповідати інтегральному індексу пошкодження.

Існують різні методи дефазифікації: центр ваги (centroid method), максимум (highest membership), середнє значення максимумів, інтервальний метод тощо. Найпоширенішим є метод центру ваги, який забезпечує згладжені, стабільні результати та добре підходить для технічних додатків.

1.3.2. Архітектура та принцип роботи системи нечіткого виведення

З практичної точки зору, реалізація нечіткої логіки відбувається у вигляді системи нечіткого виведення (Fuzzy Inference System, FIS) [19]. Це набір формалізованих компонентів, які поєднуються для обробки вхідних даних, логічного аналізу на основі заданих правил та генерації результату (рисунок 1.2).

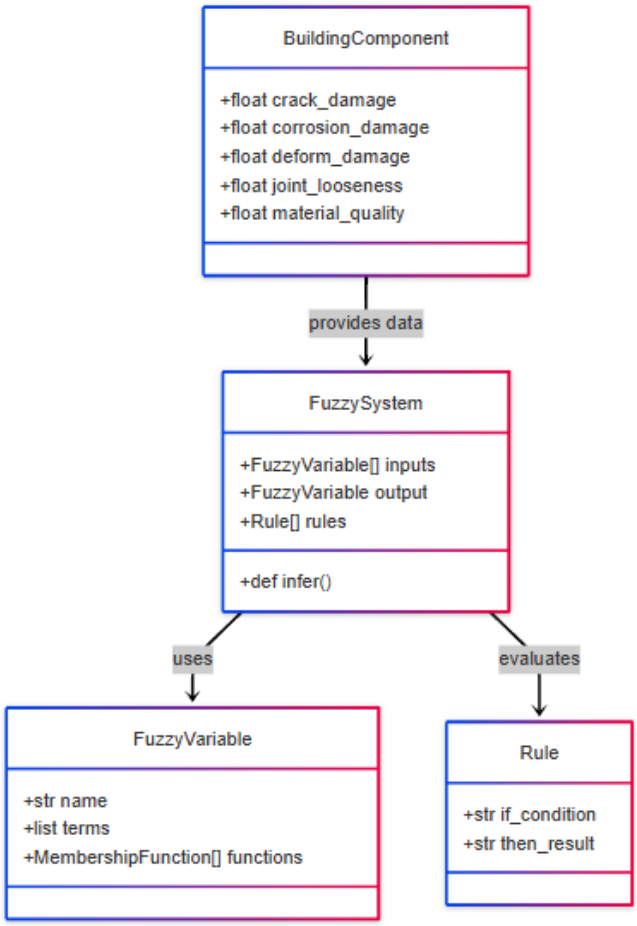


Рисунок 1.2. Діаграма класів

Найбільш поширеною і зрозумілою є архітектура Мамдані, яка має таку структуру:

- Фазифікатор: перетворює вхідні значення у ступені приналежності;
- База правил: складається з тверджень типу «якщо… тоді…» (if-then);
- Механізм логічного виведення: комбінує значення за допомогою логічних операцій (AND, OR, NOT);
- Агрегатор: об’єднує результати всіх правил у єдину вихідну нечітку множину;
- Дефазифікатор: перетворює результат у числову оцінку.

Приклад правила в нечіткій системі може виглядати так:

- Якщо тріщини великі і якість матеріалу низька, то рівень пошкодження — високий.

Таблиця 1.2

Типові правила нечіткого виведення

Правило (if-then)	Категорія результату
Якщо crack = low і corrosion = low, тоді damage = minor	Minor
Якщо crack = medium і deform = medium, тоді damage = moderate	Moderate
Якщо crack = high і material = low, тоді damage = significant	Significant
Якщо joint_looseness = high і corrosion = high, тоді damage = significant	Significant
Якщо crack = medium і material = medium, тоді damage = moderate	Moderate

Важливо, що такі правила активуються частково — не в режимі "активне/неактивне", а з відповідною інтенсивністю, що забезпечує гладкість і плавність переходів між класами.

Операції AND, OR і NOT мають специфічну реалізацію:

- AND: min $(\mu_1\mu_2)$ — мінімальне з двох значень;
- OR: max $(\mu_1\mu_2)$ — максимальне з двох;

- NOT $1 - \mu$: — доповнення до одиниці.

Ці правила дозволяють моделювати взаємозв’язки між параметрами без потреби в точному числовому рівнянні. В результаті система здатна генерувати обґрунтовані оцінки навіть тоді, коли частина вхідної інформації є неповною або наближеною.

1.3.3. Застосування в будівельній інженерії

Однією з важливих галузей застосування нечіткої логіки є будівництво та технічна експертиза, де через складність об'єктів, різноманітність впливів та природну невизначеність дефектів часто складно побудувати точні аналітичні моделі.

Досвід показує, що експертна оцінка технічного стану будівлі зазвичай базується на комплексному враженні, яке формується інженером на основі візуального огляду, часткових вимірювань, інтуїції та аналогій. Саме ця інтуїтивна форма знання і є основою для створення нечіткої бази правил.

У публікаціях, присвячених темі «Fuzzy-based building damage assessment», наводяться численні приклади успішного застосування FIS у діагностиці стану мостів, багатоповерхових споруд, індустріальних об’єктів [20]. В Японії після землетрусів 2011 року було створено мобільні додатки, які дозволяли рятувальникам швидко вводити оціночні параметри стану будівель, після чого система видавала ймовірність обвалу. В Італії нечітка логіка інтегрується у BIM-моделі для формування рекомендацій щодо черговості ремонтів.

В Україні такі системи ще на етапі впровадження, але потенціал очевидний: поєднання зрозумілих інженерові правил із автоматичними розрахунками дозволяє побудувати гібридну експертно-цифрову модель, що стане потужним інструментом у руках як досвідченого фахівця, так і початківця.

1.4. Визначення об’єкта, мети та завдань проєктування

У будь-якому інженерному дослідженні першим і ключовим кроком є чітке визначення того, що саме вивчається (об’єкт дослідження), у чому полягає інноваційна складова або ракурс аналізу (предмет дослідження), якої мети передбачається досягти та які завдання необхідно вирішити на шляху до реалізації цієї мети. У рамках дипломної роботи бакалавра особливо важливо системно, логічно та послідовно сформулювати усі ці компоненти, адже саме вони є методологічною основою для всієї подальшої побудови логіки дослідження, розробки програмної частини, аналізу результатів та формування висновків.

1.4.1. Об’єкт та предмет дослідження

Об’єктом дослідження в даній роботі є будівля або її конструктивна складова, яка зазнала певного ступеня пошкодження внаслідок експлуатаційних, природних, техногенних або аварійних чинників. Під терміном "будівля" розуміється інженерно-будівельний об'єкт, який виконує функцію житлового, громадського чи промислового призначення та включає комплекс конструктивних елементів, таких як фундамент, цоколь, несучі стіни, балки перекриття, колони, плити перекриття, дахи, з'єднувальні вузли та ін.

У межах роботи основну увагу буде зосереджено на таких фрагментах конструкцій, які є критично важливими для забезпечення стійкості та безпеки будівлі. Зокрема, йдеться про:

- цокольний поверх як фундаментально навантажену частину будівлі, що піддається дії ґрунтових вод, морозу, вологи, осідання основи;
- несучі елементи конструкцій — вертикальні (стіни, колони) та горизонтальні (балки, перекриття) — як основа для передачі навантажень, стабільності споруди;
- вузли з’єднань, які часто виявляються слабкою ланкою внаслідок монтажних помилок, старіння або впливу повторюваних навантажень.

Саме такі конструктивні фрагменти найчастіше піддаються впливам, які призводять до виникнення дефектів — тріщин, корозії, деформацій, розшарувань — і саме вони потребують регулярного та об’єктивного моніторингу.

Слід наголосити, що об’єкт дослідження не обмежується фізичною спорудою як такою — він охоплює також сукупність її фізико-механічних характеристик, які можуть бути кількісно або якісно виміряні, а також змінюються з плином часу. Таким чином, дослідження ведеться не лише на рівні матеріальної структури, а й на рівні її динамічного стану, що обумовлює особливу актуальність застосування саме нечітких підходів.

Предметом дослідження виступає методологія обробки інформації про технічний стан конструкцій із використанням принципів нечіткої логіки, а також розробка відповідної обчислювальної підсистеми, яка реалізує зазначений підхід у формі цифрового програмного засобу.

Інакше кажучи, предметом дослідження є не будівля як така, а саме процес формалізації знань експерта у вигляді нечіткої бази правил, трансформації вхідної інформації (яка може бути неповною, приблизною або лінгвістично описаною) у кількісний показник технічного стану, що надалі може бути використаний для:

- прийняття рішень щодо необхідності ремонту,
- визначення черговості втручань,
- класифікації будівель за категоріями пошкодження.

Цей підхід є надзвичайно важливим у сучасних умовах, коли у сфері будівельного контролю бракує як часу, так і ресурсів для повноцінного інструментального дослідження кожного об’єкта. Саме тому предмет дослідження можна розглядати як інструмент інтелектуалізації будівельної експертизи, тобто перехід від ручного, інтуїтивного оцінювання до частково автоматизованого процесу, який базується на формалізованій, але гнучкій логіці.

1.4.2. Постановка задачі

Постановка задачі дослідження вимагає чіткого визначення як вхідних даних, які користувач повинен надати системі, так і очікуваних вихідних результатів, які мають бути обчислені на основі цих даних. У ролі користувача може виступати як експерт у галузі будівництва, так і технік або оператор, що проводить оцінювання стану об’єкта.

У межах цього дослідження як вхідні дані використовуються п’ять основних параметрів, що репрезентують найбільш суттєві характеристики пошкоджень будівельних конструкцій. Всі ці параметри задаються у відсотковій шкалі (від 0 до 100%). Такий формат є інтуїтивно зрозумілим, дозволяє формалізувати навіть приблизні візуальні оцінки без необхідності точного інструментального контролю, і тому особливо зручний для використання у польових умовах.

Першим параметром є ступінь тріщиноутворення (crack_damage), який відображає розвиток тріщин у конструкції. Це можуть бути як поверхневі тріщини, так і глибокі наскрізні пошкодження. Нульове значення свідчить про відсутність тріщин, тоді як 100% вказує на повну втрату цілісності елемента.

Другим параметром виступає ступінь корозійного ураження (corrosion_damage). Йдеться насамперед про армовані бетонні та металеві елементи конструкцій. Чим більшим є значення цього параметра, тим інтенсивнішими є процеси втрати перерізу арматури, спучення бетону, руйнування зв’язків між елементами.

Третій показник — рівень деформацій (deform_damage) — відображає наявність геометричних відхилень, прогинів, перекосів, зсувів, які можуть свідчити про втрату стійкості елементів або локальні пошкодження основи. При високих значеннях параметра йдеться про суттєву втрату геометричної стабільності.

Четвертим параметром є ступінь розхитування з’єднань (joint_looseness), що фіксує стан анкерних, болтових, зварних та інших типів з’єднань у конструкції. Навіть помірне ослаблення таких вузлів суттєво знижує загальну несучу здатність споруди, особливо в умовах циклічного або динамічного навантаження.

П’ятий параметр — це якість матеріалу (material_quality). Це інтегральна характеристика, що відображає загальний стан будівельного матеріалу (бетону, цегли, дерева, металу) з урахуванням таких факторів, як старіння, дія вологи, втома, хімічна деградація. Чим нижче значення цього параметра, тим гіршою є залишкова здатність матеріалу виконувати несучу функцію.

Разом ці п’ять параметрів охоплюють ключові типи пошкоджень, що характерні для реальних об’єктів. Такий набір дозволяє з мінімальним обсягом

вхідної інформації отримати достовірну оцінку технічного стану без потреби в десятках вузькоспеціалізованих показників. Це робить модель придатною для широкого застосування в різних умовах, зокрема й у ситуаціях обмеженого доступу до обладнання чи документації.

Що стосується вихідної інформації, то система повинна повертати користувачеві два основні результати. Перший — це індекс пошкодження (Damage Index), тобто числове значення, що лежить у межах від 0 до 1. Воно є результатом математичної процедури дефазифікації та виступає інтегральною оцінкою технічного стану елемента або всієї конструкції. Другий результат — це категорія пошкодження (Damage Level), що є якісною інтерпретацією отриманого числового індексу. У рамках розроблюваної підсистеми передбачається поділ на три основні категорії:

- Minor — незначні пошкодження (індекс у діапазоні 0.0–0.3),
- Moderate — помірні пошкодження (0.3–0.7),
- Significant — значні пошкодження (0.7–1.0).

Таке групування дає змогу не лише кількісно оцінити технічний стан, а й приймати управлінські рішення — наприклад, про доцільність ремонту, подальшого моніторингу або виведення конструкції з експлуатації.

Завдання проєктування, яке реалізується в рамках цієї дипломної роботи, полягає у створенні інтелектуальної підсистеми, що:

- сприймає п’ять вхідних характеристик пошкоджень, виражених у відсотках;
- опрацьовує їх за допомогою механізмів нечіткої логіки, зокрема фазифікації, бази правил, агрегування та дефазифікації;
- видає зрозумілий і обґрунтований результат — індекс пошкодження та відповідну категорію, які можуть бути використані для подальших інженерних дій.

Таким чином, система поєднує переваги експертного підходу з алгоритмічною точністю, що забезпечує ефективність її застосування в реальній інженерній практиці.

Розділ 2. СПЕЦИФІКАЦІЯ ВИМОГ, МЕТОДИ І МОДЕЛІ

2.1. Вимоги до підсистеми оцінки пошкоджень

Проєктування підсистеми оцінки пошкоджень будівель базується на результатах попереднього аналітичного розділу, де було сформульовано концептуальне бачення проблеми, обґрунтовано доцільність використання нечіткої логіки, визначено набір вхідних параметрів і структуру результатів. У цьому підрозділі здійснюється формалізація вимог до розроблюваної підсистеми — як з позиції функціональності, так і з урахуванням програмно-технічних обмежень.

2.1.1. Загальні вимоги

Підсистема повинна бути реалізована у форматі web-додатку, що дозволяє забезпечити її доступність з будь-якого пристрою без потреби локальної інсталяції. Вибір середовища Dash (Python) зумовлений його інтегрованими можливостями для побудови інтерактивних інтерфейсів, графіків і роботи з серверною логікою.

Вхідні дані задаються у зручному для користувача вигляді — у відсотковій формі (від 0 до 100%), що полегшує їх інтерпретацію. Однак у внутрішній логіці підсистеми всі значення повинні бути автоматично нормалізовані у шкалу від 0 до 1, яка є стандартною у нечіткій логіці та дозволяє коректно обчислювати значення функцій приналежності.

Окремим положенням встановлюється вимога щодо збереження результатів кожного сеансу оцінювання. З цією метою у структурі системи має бути реалізований зв’язок з базою даних SQLite, що забезпечить локальне зберігання ключових параметрів: ідентифікатора об’єкта, дати та часу оцінювання, значення індексу пошкодження та відповідної категорії.

2.1.2. Функціональні вимоги

У структурі системи повинна бути реалізована низка функціональних компонентів, які забезпечують повний цикл: від введення даних до генерації результатів і збереження інформації.

Передусім користувач має змогу задати ідентифікатор об’єкта, який дозволяє однозначно пов’язати введені параметри з конкретною будівлею або її фрагментом. Для введення основних технічних параметрів використовуються п’ять слайдерів, кожен із яких відповідає за окрему ознаку: ступінь тріщин, корозії, деформацій, розхитування з’єднань і якість матеріалу.

Після задання вхідних значень система повинна генерувати графік функцій приналежності для кожного параметра. Ці графіки змінюються динамічно — у режимі реального часу — залежно від положення відповідного слайдера. Таким чином, користувач отримує наочне уявлення про те, до яких нечітких множин (low, medium, high) потрапляє кожне значення.

Ключовою дією є запуск обчислення нечіткого виведення за допомогою кнопки «Запустити оцінку». У відповідь система виконує фазифікацію вхідних даних, активує відповідні правила з бази знань, виконує агрегування результатів і проводить дефазифікацію для отримання фінального числового індексу пошкодження.

Результати виводяться у кількох формах. Перша — це таблиця зі ступенями приналежності (μ): для кожного параметра виводяться значення μ для трьох термів (низький, середній, високий). Це дозволяє користувачеві побачити, які множини активовані найбільше.

Другою формою подання є графік агрегованої нечіткої функції, на якому показано як форму об’єднаного результату, так і вертикальну лінію дефазифікації, яка вказує на отриманий числовий індекс.

Також система повинна формувати текстовий висновок, у якому чітко вказується значення Damage_Index та відповідна категорія пошкоджень — Minor (незначні), Moderate (помірні) або Significant (значні). Цей висновок є основою для прийняття інженерних рішень.

Окрім візуального відображення, користувач має змогу завантажити результати у форматі CSV або згенерувати HTML-звіт, який міститиме як числові значення, так і графіки. Звіти можуть бути використані для формування актів обстеження або архівування.

Насамкінець, система повинна автоматично зберігати результати в базу даних SQLite, де фіксується ідентифікатор об’єкта, дата й час сеансу, обчислене значення Damage_Index та обрана категорія. Це забезпечує накопичення статистики, можливість повторного аналізу та формування звітності.

2.1.3. Функціональні вимоги

З погляду нефункціональних характеристик система має відповідати критеріям зручності, швидкодії, стабільності та захищеності.

Інтерфейс має бути інтуїтивно зрозумілим, компактним та адаптованим до стандартних роздільних здатностей екрана, зокрема максимум до 1024×768 пікселів, що дозволяє використовувати систему на планшетах або ноутбуках із середніми технічними характеристиками.

Час виконання основної логіки — тобто обчислення нечіткого виведення після натиснення кнопки — не повинен перевищувати 1–2 секунди, навіть у разі багаторазового повторення операцій.

Для запобігання помилкам має бути реалізовано захист від некоректного введення: поле «Ідентифікатор об’єкта» повинно приймати лише текстові значення, тоді як усі слайдери — лише числа в діапазоні від 0 до 100.

З технічного боку, код програми повинен відповідати стандартам PEP8 щодо чистоти й читаємості, а також містити зрозуміле коментування всіх функцій у вигляді докстрингів, що полегшить супровід, модифікацію та перевірку системи.

2.2. Формалізація вхідних даних і нечіткого моделювання

Процес створення підсистеми оцінки технічного стану будівель на основі нечіткої логіки передбачає не лише програмну реалізацію алгоритмів, а й

математичну формалізацію як вхідних параметрів, так і процедури нечіткого виведення. Даний підрозділ присвячено опису методів нормалізації даних, визначенню функцій приналежності для кожної ознаки, побудові нечітких множин, а також специфікації вихідних лейблів системи.

Уся вхідна інформація, яку користувач вводить у підсистему, подається у вигляді відсоткових значень у діапазоні від 0 до 100%. Такий підхід є зручним для інтерфейсної взаємодії, особливо в контексті застосування слайдерів та суб'єктивної оцінки фахівця. Однак для коректної роботи нечіткого механізму виведення всі вхідні значення мають бути приведені до нормалізованого діапазону [0; 1], що відповідає загальноприйнятій практиці у нечіткій логіці.

Нормалізація здійснюється за лінійною формулою:

$$x_{\text{норм}} = \frac{x\%}{100} \tag{2}$$

де

$x\%$ - введене користувачем значення у відсотках

$x_{\text{норм}}$ - нормалізоване значення, яке використовується далі для розрахунку ступенів приналежності до нечітких множин.

2.2.1. Нечіткі множини та функції приналежності вхідних параметрів

Для кожного з вхідних параметрів системи були визначені нечіткі множини за допомогою відповідних функцій приналежності (MF), які забезпечують перехід від чітких значень до лінгвістичних оцінок. Залежно від характеру змінної використовувались трапецієвидні або трикутні MF. Приклади графіків наведено на відповідних рисунках.

Для параметра ступінь тріщиноутворення (crack_damage) було побудовано три нечіткі множини. Множина *Low* реалізована за допомогою трапецієвидної функції, що охоплює значення, близькі до нуля, та моделює ситуації, коли пошкодження практично відсутнє. Значення *Medium* описується трикутною MF з піком у середньому діапазоні (приблизно від 0.4 до 0.6), що відповідає помірним тріщинам. Високий рівень пошкоджень (*High*) задається трапецієвидною MF, яка

охоплює крайні значення та моделює глибокі або критичні тріщини. Візуалізація цих функцій наведена на рисунку 2.1.

- Low — трапецієвидна MF, яка охоплює значення, близькі до нуля. Відображає ситуації, коли пошкодження не візуалізується або є мінімальним.
- Medium — трикутна MF з піком у середньому діапазоні (~0.4–0.6), яка інтерпретує помірні тріщини.
- High — трапецієвидна MF, яка охоплює крайні значення та моделює глибокі або критичні тріщини.

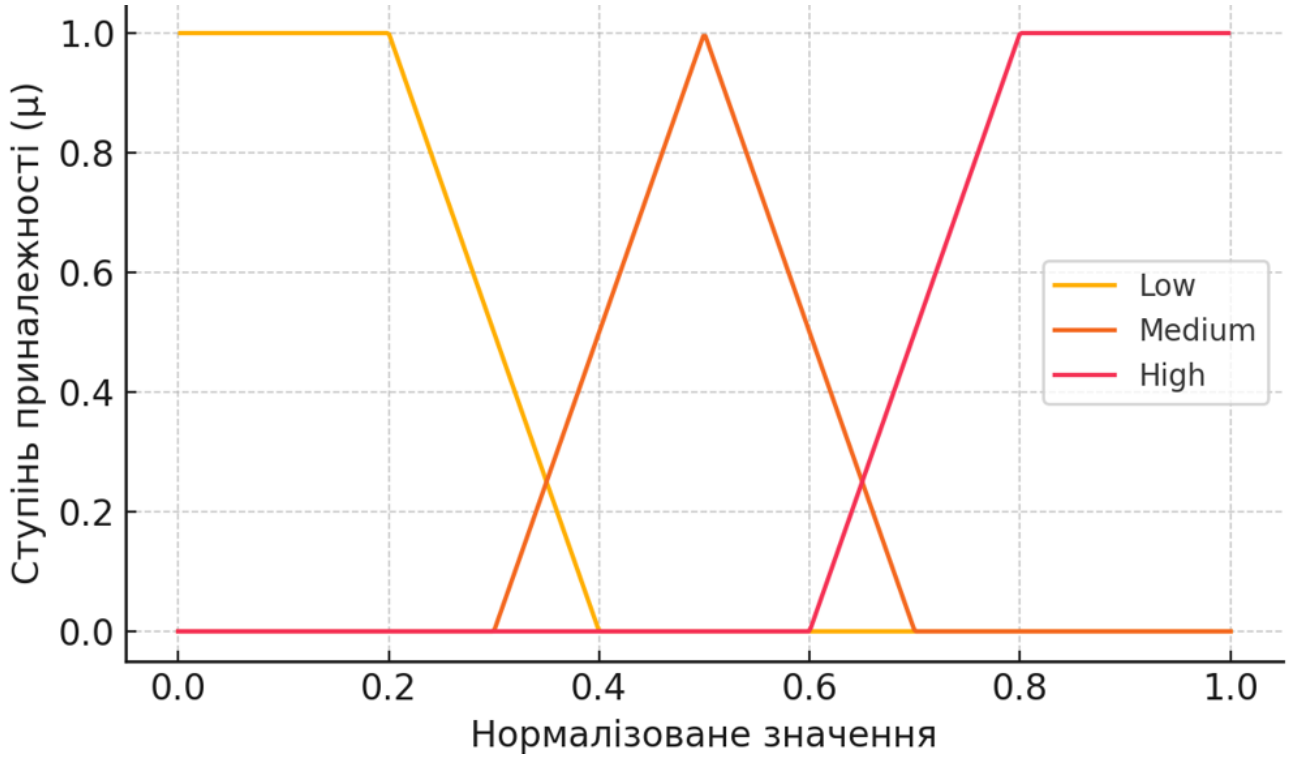


Рисунок 2.1. Функція приналежності для crack_damage

Для параметра ступінь корозійного ураження (corrosion_damage) застосовано аналогічну структуру. Множина *Low* описує поверхневу іржу без впливу на несучу здатність, *Medium* — помірне ураження металевих елементів, а *High* — глибоку корозію, що супроводжується спученням бетону та оголенням арматури. Графік функцій приналежності для цього параметра представлено на рисунку 2.2.

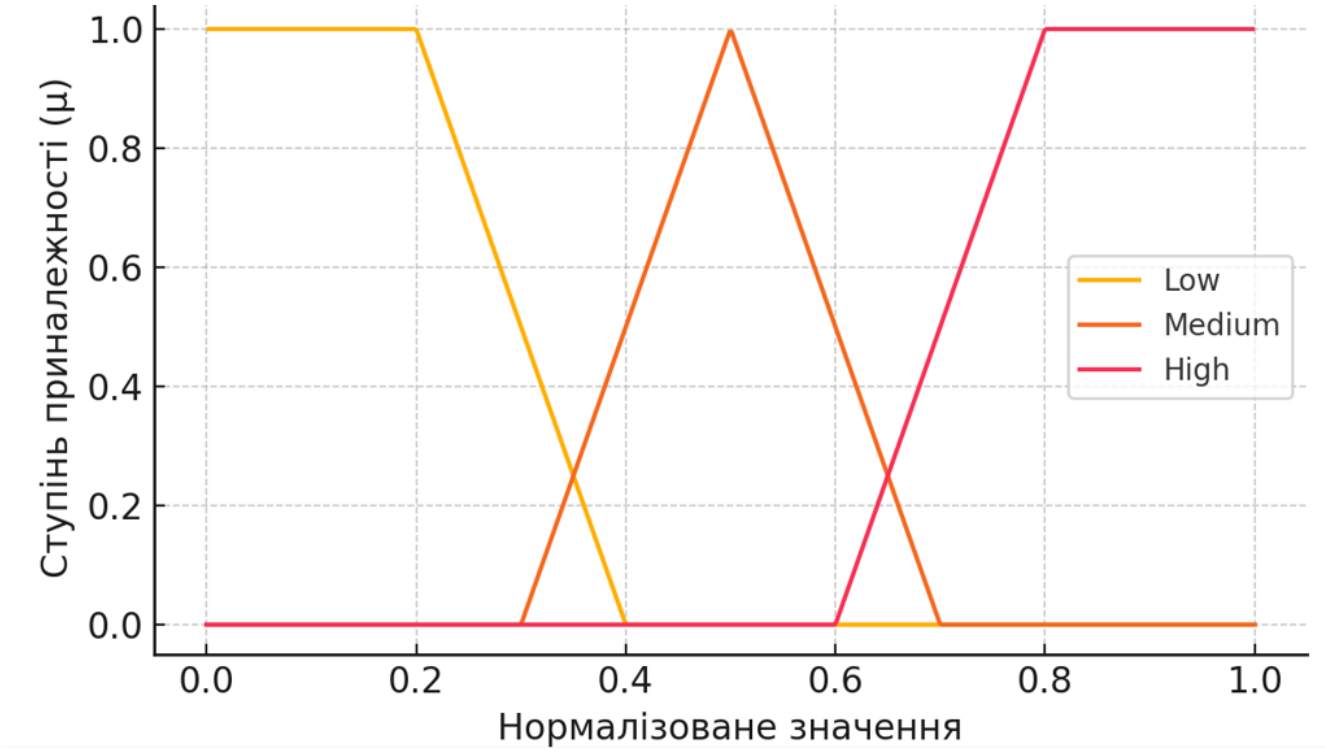


Рисунок 2.2. Функція приналежності для corrosion_damage

- Low — трапецієвидна MF, що відповідає слабкому прояву іржі на поверхні без ураження конструктивної здатності.
- Medium — трикутна MF, яка описує помірне ураження металу або арматури.
- High — трапецієвидна MF, яка моделює ситуації з глибокою корозією, спученням бетону та оголенням арматури.

У випадку параметра рівень деформацій (deform_damage) множина *Low* відповідає нетиповим, але допустимим відхиленням, *Medium* — контрольованим прогинам, перекосам або локальним зсувам, а *High* — суттєвим деформаціям, що можуть свідчити про критичне зниження геометричної стабільності. Графічне зображення функцій наведено на рисунку 2.3.

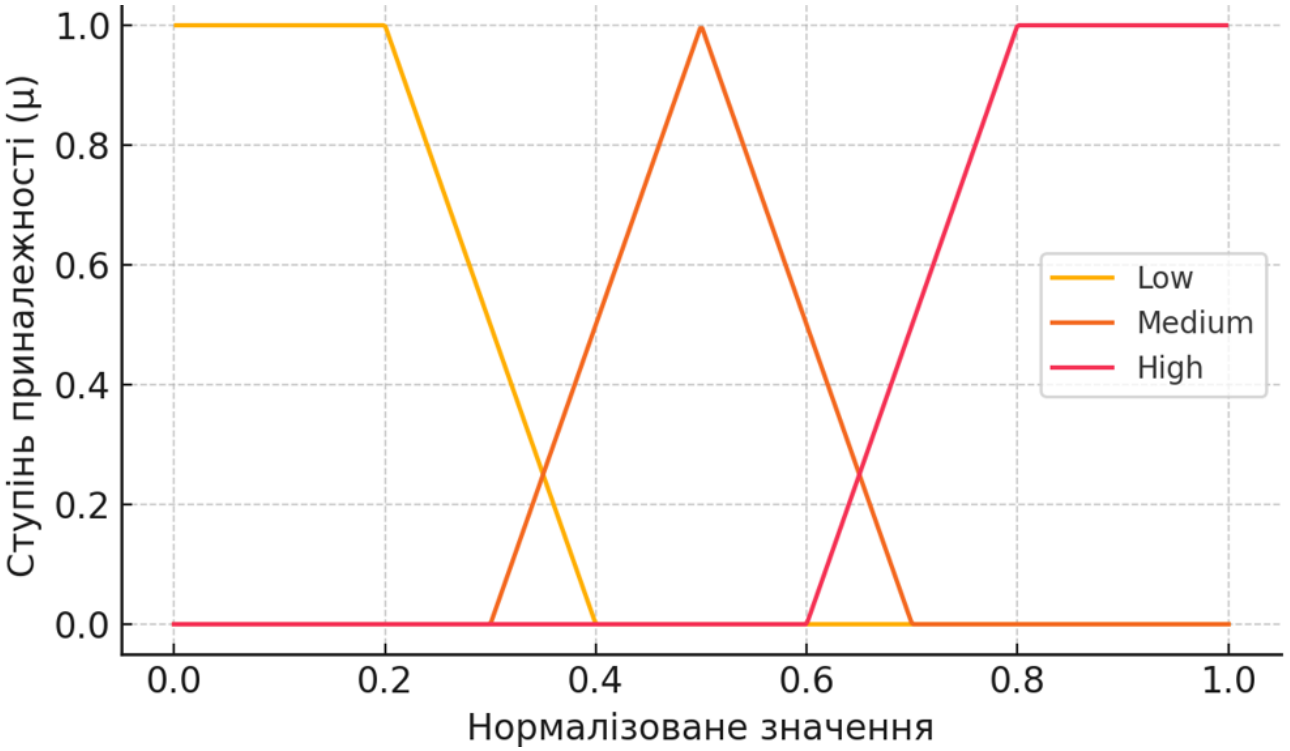


Рисунок 2.3. Функція приналежності для deform_damage

- Low — трапецієвидна MF, що охоплює нетипові, але не критичні деформації.
- Medium — трикутна MF, що описує контрольовані прогини, помірні перекоси.
- High — трапецієвидна MF, що включає небезпечні деформації з потенційною загрозою стабільності.

Для параметра розхитування з’єднань (joint_looseness) також використано три рівні. *Low* сигналізує про повну надійність з’єднань, *Medium* описує початкове розхитування, а *High* — значну втрату фіксації вузлів, що може призводити до локальних руйнувань. Функції приналежності зображено на рисунку 2.4.

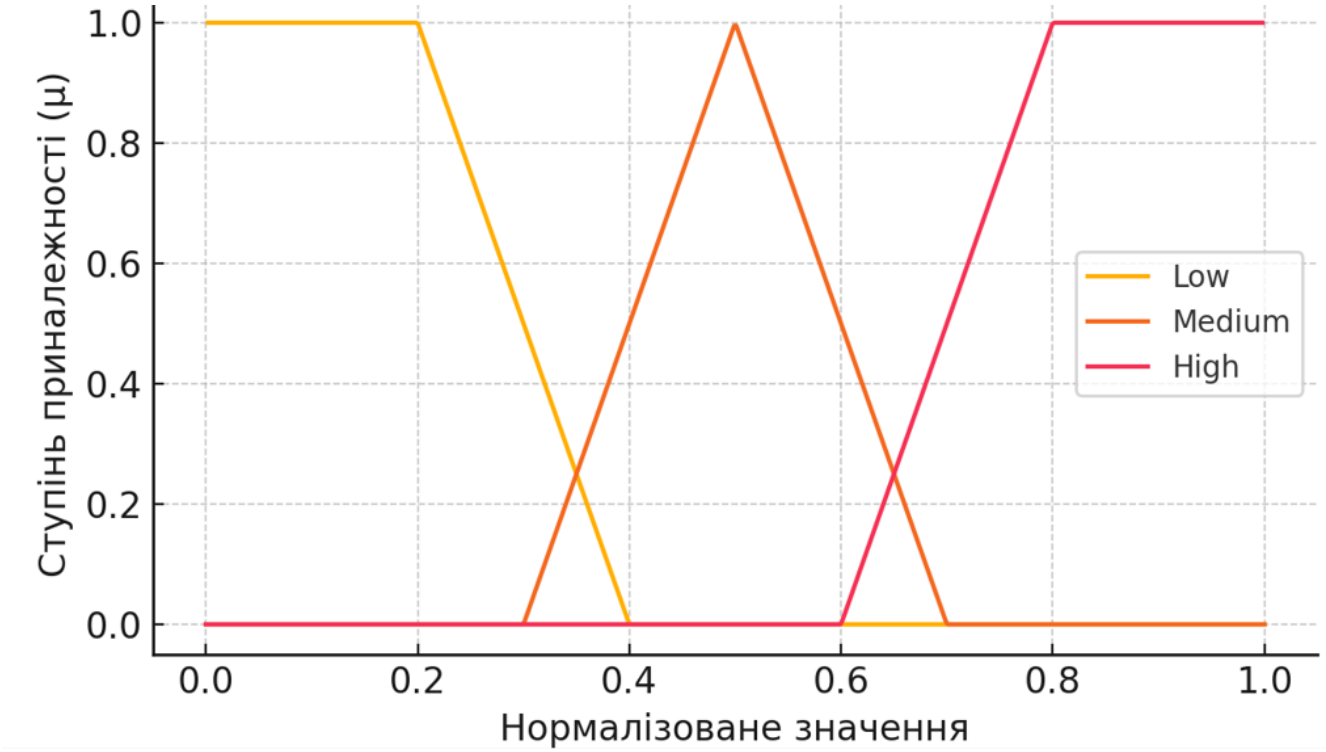


Рисунок 2.4. Функція приналежності для joint_looseness

- Low — трапецієвидна MF, яка сигналізує про надійний стан з’єднань.
- Medium — трикутна MF, що описує початкове розхитування або люфти в стиках.
- High — трапецієвидна MF, яка інтерпретується як істотна втрата з’єднання елементів.

Останнім вхідним параметром є якість матеріалу (material_quality). Оскільки зниження якості є негативним фактором, для цього параметра використовується зворотна інтерпретація шкали. Значення High відповідає доброму стану матеріалу та задається трапецієвидною MF. Стан Medium — це середня якість з початковими мікротріщинами або ознаками деградації, а Low — значне зниження міцності, крихкість, хімічне ураження. Відповідні функції приналежності подані на рисунку 2.5.

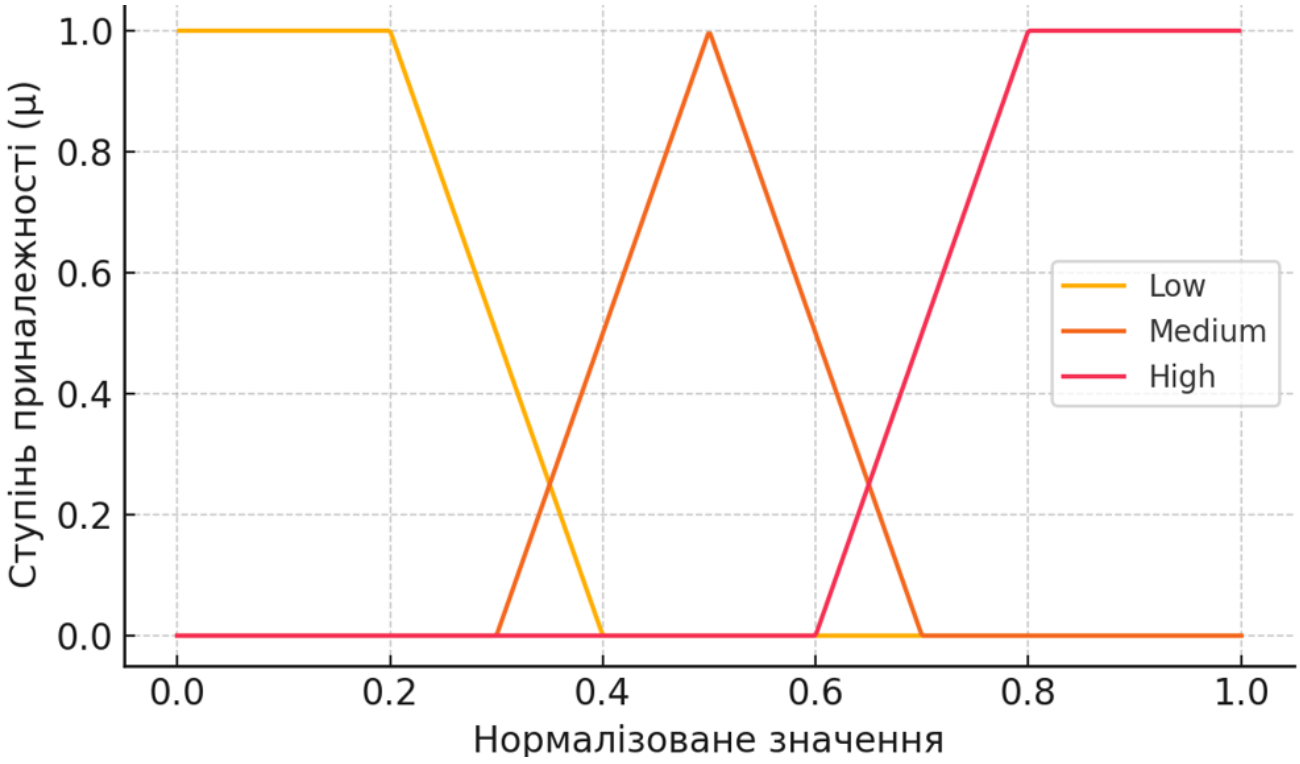


Рисунок 2.5. Функція приналежності для material_quality

Опис вихідної змінної (Damage_Level) та її лейблів подано окремо в наступному пункті. Графіки нечітких множин результату наведено на рисунку 2.6.

- High — трапецієвидна MF з високими значеннями, що вказує на добрий стан матеріалу.
- Medium — трикутна MF, яка охоплює середній рівень із початковими проявами деградації.
- Low — трапецієвидна MF, що відображає значне зниження міцності, мікротріщини, крихкість.

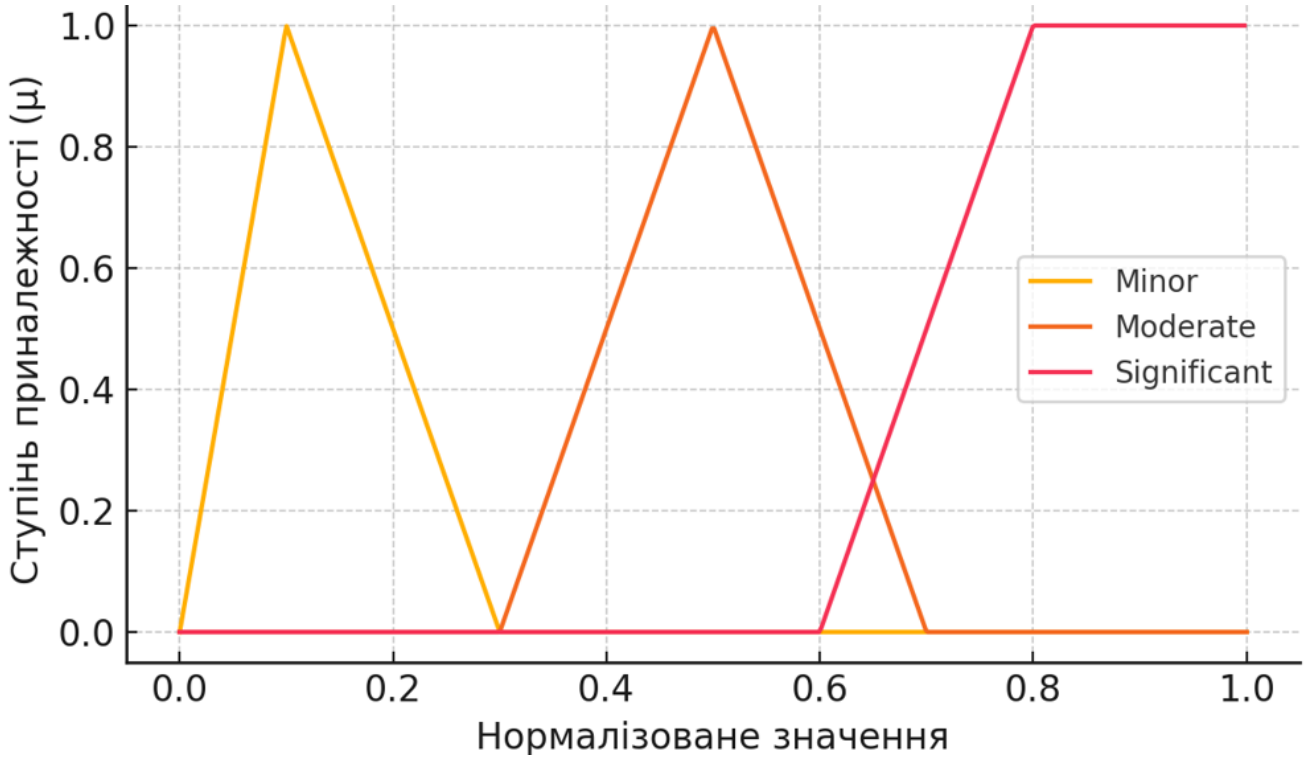


Рисунок 2.6. Функція приналежності для Damage_Level

2.3. Формування бази правил

У підсистемі оцінки технічного стану конструкцій ключовим етапом є побудова **бази нечітких правил**, що відображає знання досвідчених інженерів щодо поєднання різних ознак пошкоджень. Ці правила описуються у форматі «якщо … то …» та складають ядро механізму нечіткого виведення Мамдані, де кожна умова («якщо…») зв’язана з одним або кількома вхідними параметрами, а висновок («то…») описує лінгвістичну категорію вихідного показника Damage_Level.

Правила формувалися експертами шляхом систематичного аналізу реальних випадків обстеження будівельних конструкцій. Для кожного з п’яти вхідних параметрів — ступеня тріщин, рівня корозії, величини деформацій, розхитування вузлів та якості матеріалу — виділено три нечіткі категорії: low, medium, high. Далі експерти обирали ті поєднання цих категорій, що найчастіше зустрічаються в реальній практиці, і призначали їм одну з трьох можливих кінцевих оцінок: Minor, Moderate або Significant.

У підсистемі закладено три основні правила, які ілюструють типові сценарії:

- Критичне поєднання дефектів: коли тріщини виражені сильно, корозія помірна, деформації та розхитування вузлів мінімальні, а якість матеріалу залишається високою, результат інтерпретується як Significant. У цьому випадку наявність глибинних тріщин у поєднанні з прогресуючою корозією призводить до серйозного зниження несучої здатності.
- Узагальнена середня ситуація: якщо всі п’ять параметрів перебувають у середньому діапазоні нечіткості (medium), це свідчить про помірні пошкодження (Moderate). Такий комплексний стан характерний для об’єктів в експлуатації середнього віку, де одночасно виявляються різні незначні дефекти.
- Незначні відхилення: коли інтенсивність усіх пошкоджень є низькою, а матеріал зберігає високу якість, рівень пошкодження оцінюється як Minor. Такий сценарій відповідає обстеженням нових або недавно відремонтованих споруд, у яких дефекти практично відсутні.

Ці приклади правил не вичерпують усіх можливих комбінацій, але забезпечують репрезентативний базис. При необхідності база може бути розширена ще кількома правилами для покриття специфічних випадків (максимум 5–7), не ускладнюючи логіку та зберігаючи простоту підтримки.

Після того як під час сеансу оцінювання активуються всі правила, кожне з них продукує нечітке висновкове множинне значення з інтенсивністю, що визначається мінімумом ступенів приналежності по умові кожного терму. Далі всі ці нечіткі множини об’єднуються операцією максимуму, що дає змогу отримати єдину агреговану нечітку функцію на просторах вихідної змінної. Цей підхід гарантує, що жодне з правил не випаде з оцінки, а найбільш «потужні» поєднання визначатимуть форму об’єднаної множини.

Для перетворення цієї агрегованої нечіткої множини на чітке число використовується метод центра ваги (centroid). Він полягає у знаходженні центру мас побудованої поверхні. При дискретизації області визначення (наприклад, 101 точка на відрізку від 0 до 1) цей метод демонструє швидку й досить точну роботу,

що забезпечує швидкість обчислень менше однієї секунди навіть на звичайному ПК.

Вибір архітектури Мамдані для нечіткого виведення зумовлений її інтуїтивною зрозумілістю для фахівців з будівельної діагностики: вони легко можуть безпосередньо формулювати правила в термінах «якщо … то …». Альтернативний підхід Сугено передбачає формули в висновку, що робить систему менш прозорою для інженера, який звик оперувати саме лінгвістичними категоріями.

Використання трикутних та трапецієвидних функцій приналежності обрано з огляду на два фактори: по-перше, такі MF забезпечують достатню гнучкість для моделювання широкого спектру фізичних станів; по-друге, їхня обчислювальна простота дозволяє легко інтегрувати систему у веб-додаток із реальним часом відгуку.

Отже, сформована база правил із трьома ключовими сценаріями, класичною агрегацією–дефазифікацією за Мамдані та простою структурою MF становить оптимальний компроміс між точністю, прозорістю та швидкодією.

2.4. Архітектура програмної підсистеми

На рисунку 2.7 ілюструється логічний потік даних: користувач вводить ідентифікатор та п’ять параметрів через поля та слайдери, ці значення проходять фазифікацію й нечітке виведення, потім відбувається дефазифікація вхідного нечіткого множення до числового Damage_Index, а результат відображається в таблиці, графіку та тексті. За потреби той же набір даних зберігається в SQLite і експортується у CSV або HTML.

Рисунок 2.7. Ппідсистема оцінки пошкоджень будівель

2.4.1. Безпека та контроль доступу

Незважаючи на відносну простоту архітектури, будь-яка веб-підсистема, що обробляє дані користувача, потребує базових механізмів захисту. По-перше, усі поля введення проходять валідацію на стороні сервера, щоб запобігти ін'єкціям SQL та передати в базу лише чисті, очікувані значення. object_id проходить перевірку на наявність лише буквено-цифрових символів, а п'ять слайдерів — обмежені строго діапазоном 0–100.

По-друге, взаємодія з базою SQLite відбувається через підготовлені запити з параметризованими плейсхолдерами, що практично повністю виключає ризик витоку або спотворення даних. Архітектура передбачає можливість додати рівень аутентифікації або авторизації (наприклад, за допомогою JWT-токенів), якщо

підсистему вирішать відкривати не лише внутрішнім інженерам, а й широкому колу користувачів.

Нарешті, у виробничому середовищі рекомендовано розгорнути подачу через HTTPS (сертифікат Let’s Encrypt або корпоративний CA) і захистити бекенд за допомогою зворотного проксі (NGINX чи Apache), який обмежить прямий доступ до скриптів Python і вбереже від DDOS-атак первинним фільтром на рівні мережі.

2.4.2. Розгортання та масштабування

Для невеликих команд найзручнішим способом розгортання є контейнеризація: створення Docker Image, що містить усі залежності (requirements.txt), конфігурацію config.py та готовий веб-сервер Gunicorn (або Uvicorn із ASGI-шлюзом). Таке рішення дозволяє один раз підготувати образ і запускати його скрізь — на локальній машині, у приватному дата-центрі або в хмарі.

В умовах зростання навантаження можна горизонтально масштабувати кілька екземплярів контейнера за допомогою Kubernetes, Docker Swarm чи навіть AWS ECS. Балансування запитів здійснюється через стандартні інструменти (Ingress-контролер, ELB), що дозволяють підтримувати обслуговування сотень одночасних користувачів із середнім часом відповіді в межах 200–300 мс.

У разі необхідності зберігати великий обсяг історичних даних варто перенести SQLite у більш «промисловий» варіант (PostgreSQL, MySQL) — це не вимагатиме переписування логіки самого Python-модуля data_manager, а лише налаштування нового підключення в config.py.

2.4.3. Моніторинг, логування та підтримка

Працююча підсистема не повинна бути «чорною скринькою» — з моменту запуску важливо фіксувати всі ключові події й мати змогу аналізувати їх у разі непередбачених збоїв.

У коді реалізовано логування через стандартний модуль logging, причому кожен callback-обробник генерує записи рівнів INFO (старт оцінки, кінцевий

результат), WARNING (нетипова поведінка, наприклад, порожній object_id) і ERROR (несподівані винятки при зверненні до БД або бібліотек).

Для оперативного виявлення деградації швидкості роботи варто інтегрувати звукове або візуальне сповіщення через Sentry чи Prometheus: Sentry збирає повний стек-трек помилок, а Prometheus у парі з Grafana відслідковує час обробки HTTP-запитів і завантаження CPU/пам’яті.

Регулярні backup файлів бази SQLite (щодня о неробочому часі) страхують від втрати даних. А через просту конфігурацію config.py можна змінювати частоту збереження, шляхи до лог-файлів та інші «дрібниці» без необхідності перезапускати весь додаток.

Розділ 3. РЕАЛІЗАЦІЯ ТА ПРАКТИЧНА АПРОБАЦІЯ ПРОГРАМНОЇ ПІДСИСТЕМИ

3.1. Загальні принципи побудови коду

Уся підсистема побудована з урахуванням принципів «чистої архітектури», коли кожен модуль несе лише одну відповідальність, але водночас легко взаємодіє з іншими завдяки чітко описаним інтерфейсам. Ми відразу обрали сучасний інструментарій: Python 3.10 як основу, Dash для фронтенду, а весь рядок залежностей занесли в окремий файл requirements.txt (або, за бажанням, в pyproject.toml, якщо використовуєте Poetry).

Що стосується стилю коду, ключовим орієнтиром було суворе дотримання PEP 8: симетричні відступи, максимальна довжина рядка — 79 символів, імена змінних у snake_case, класи в CamelCase. Одночасно ми скористалися можливостями Python 3.10: позначили типи аргументів і результатів функцій через синтаксис def foo(x: int) -> float:, що значно полегшує читання підказок IDE і перевірку за допомогою mypy.

У проєкті передбачено автоматичне форматування коду за допомогою Black і сортування імпортів через isort. Тому як тільки файл збережено, він миттєво переформатується в єдиний стиль. Це означає, що жоден розробник не витрачатиме час на дискусії про стиль — всі pull-request'и проходять через hook, що перевіряє Black, Pylint і flake8. Таким чином, перед тим, як нова функція потрапить у репозиторій, вона вже задовольняє всі вимоги до чистоти коду.

Для організації проєкту ми віддали перевагу структурі, де папка app/ містить лише те, що стосується інтерфейсу, а пакет fuzzy/ — лише обчислення. Такий поділ не раз вигравав час: коли довелося додавати підтримку REST через FastAPI, достатньо було під'єднати той самий інтерфейс функції evaluate() з-модуля inference_engine.py, і нічого в коді Dash чіпати не довелося.

У кожному модулі присутній докладний docstring із шаблоном Google-style або NumPy-style, де позначено параметри, їхній тип, діапазон і очікуваний

результат. Завдяки цьому при відкритті функції в PyCharm (див. Рисунок 3.1) спливають підказки, які пояснюють, що приймає на вхід evaluate() і що повертає.

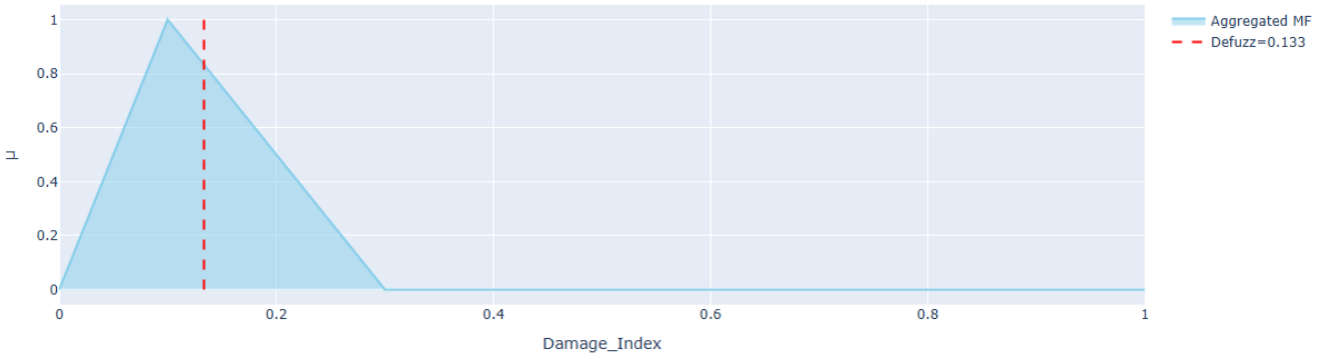


Рисунок 3.1. Автодокументація в PyCharm завдяки докстрингам і типізації

Усередині обчислювального ядра приділили увагу обробці винятків: у разі несподіваного значення або збою бібліотеки scikit-fuzzy залежно від контексту повертається або дефолтне значення індексу (наприклад, 0), або помилка логується із рівнем ERROR разом із стек-треком. Логи записуються в окремий файл logs/app.log із ротацією за розміром і кількістю днів, щоб історія помилок завжди була під рукою, але не забивала диск.

Процес тестування охоплює як класичні unit-тести за допомогою pytest, так і інтеграційні сценарії, які піднімають локальний сервер Dash у тлі, посилають HTTP-запити до припустимих шляхів і перевіряють, що JSON-відповідь містить правильний індекс. Конфігурація CI (GitHub Actions або GitLab CI) автоматично проганяє ці тести на кожен push, формуючи звіт із покриттям коду.

Для керування залежностями зберігаємо точні версії бібліотек у requirements.txt, а в описі проєкту вказано, що підтримуються Python 3.10–3.11. Усі налаштування (*thresholds*, шляхи до бази, параметри логування) винесені в config.py та можуть бути змінені через змінні середовища або .env-файл за допомогою python-dotenv.

Нарешті, проєкт пакетовано за стандартом setuptools: є setup.py, який дозволяє встановити нашу підсистему як pip-пакет, при цьому CLI-утиліта run-eval додається в PATH і дозволяє запускати оцінку з терміналу без зайвих рухів.

Завдяки такій архітектурі і практикам «DevOps-friendly» кожен фрагмент коду легко підтримувати, докладати нові тести і запускати в різних середовищах: локально через venv, у Docker-контейнері або в хмарному середовищі з Kubernetes.

3.2. Інтерфейс користувача

Веб-інтерфейс підсистеми реалізовано як єдина інтерактивна сторінка, на якій користувач може в комфорті працювати з усіма етапами нечіткого виведення та одразу ж бачити результат своїх змін. Основна ідея полягала в тому, щоб розділити екран на дві функціональні зони — ліву, призначену для введення даних, та праву, де виводяться всі проміжні й підсумкові результати.

У лівому блоці розміщено текстове поле для введення «Ідентифікатора об’єкта» і п’ять слайдерів, що відповідають за параметри crack_damage, corrosion_damage, deform_damage, joint_looseness і material_quality. Сам простір слід розглядати як єдину панель контролю: при поводженні повзунка одразу спрацьовує callback-функція, яка малює локальний графік функцій приналежності для цієї ознаки. Фрагмент коду, що відповідає за цю реакцію, знаходиться у файлі app/callbacks.py і має такий вигляд (рисунок 3.2):

```
@app.callback(
    Output('mf-graph-crack', 'figure'),
    Input('slider-crack', 'value')
)
def update_crack_mf(value):
    x = config.X_VALS
    low = fuzz.trapmf(x, [0, 0, 20, 40])
    med = fuzz.trimf(x, [30, 50, 70])
    high = fuzz.trapmf(x, [60, 80, 100, 100])

    return fig
```

Рисунок 3.2. функція оновлення локального графіка функцій приналежності

Інтерфейс користувача розроблено з прицілом на максимально інтуїтивну роботу інженера-експерта або техніка, який не обов’язково має глибокі знання в галузі веб-розробки. Сторінка складається з двох логічних зон, чітко розділених візуально, але пов’язаних єдиною схемою взаємодії. Зліва розташовані елементи введення, справа — візуалізація та підсумкові дані.

У верхній частині лівої колонки знаходиться одне текстове поле, призначене для введення унікального ідентифікатора об’єкта. Одразу під ним іде п’ять повзунків (слайдерів), кожен із яких відповідає за один із процентних параметрів: ступінь тріщин, інтенсивність корозії, величину деформацій, стан з’єднань та якість матеріалу. Завдяки тому, що повзунки мають відразу відгук — побудова локального графіка нечітких функцій приналежності — користувачу не потрібно відкривати додаткові вікна або натискати кнопки. Наприклад, легке переміщення повзунка для тріщин одразу відображає, як змінюється «трикутник» і «трапеція» на осі ступеня приналежності. Це дає змогу на льоту коригувати оцінку та інтуїтивно бачити, у яких зонах нечітких наборів знаходяться поточні значення.

Праворуч від панелі введення сконцентровано області відображення результатів. Насамперед це табличний огляд μ-значень: після того, як користувач підтвердить запит на оцінювання, у таблиці з’являються п’ять рядків і три стовпчики, де відображаються ступені належності кожної ознаки до категорій low, medium і high Такий формат дозволяє швидко порівняти показники один з одним, наприклад, побачити, що корозія перебуває на середньому рівні, а деформації — на низькому.

Нижче таблиці розміщено два графіки. Перший — це агрегована нечітка функція для вихідної змінної Damage_Level, де кольоровими зонами виділені діапазони “Minor”, “Moderate” та “Significant”. На цьому графіку також позначається точка дефазифікації, тобто числовий Damage Index. Така візуалізація є зручною, оскільки дозволяє миттєво оцінити, до якої з трьох категорій найтісніше прилягає результат. Поруч із графіком розміщено блок з текстовим висновком, у якому подано сам індекс пошкодження та його словесну інтерпретацію — наприклад, “Damage Index: 42.35 (Moderate)”.

Внизу сторінки передбачені засоби експорту: кнопки “Завантажити CSV” та “Завантажити HTML”, які активуються лише після остаточного розрахункового циклу. Завдяки цьому можна зберегти звіт у вигляді таблиці та графіка для подальшого аналізу або передачі колегам. Важливо, що весь інтерфейс адаптивно масштабується під широкоекранні та планшетні дисплеї (достатньо роздільної

здатності 1024×768), а CSS-стилі зосереджено в одному файлі, що спрощує зміну колірної гами або шрифтового оформлення.

Отже, користувацький інтерфейс підсистеми поєднує в собі простоту введення даних, моментальну візуалізацію локальних нечітких множин, зрозумілий табличний огляд і яскраву демонстрацію підсумкових показників. Завдяки такій архітектурі будь-який інженер зможе без особливих зусиль оцінити стан конструкції та зберегти результати своїх досліджень.

3.3. Модуль нечіткого виведення

Модуль нечіткого виведення є ядром інтелектуальної логіки, на якій базується вся система оцінювання технічного стану будівель. Він реалізований у вигляді ізольованого пакету fuzzy, що об’єднує декілька взаємопов’язаних компонентів. Така структура дозволяє забезпечити не тільки гнучкість у налаштуванні логіки, а й можливість легкої модифікації правил, форм функцій, а також швидке масштабування системи під інші ознаки або категорії.

Ключова ідея побудови цього модуля полягає в тому, щоб максимально наблизити логіку системи до людського експертного мислення, але при цьому залишити всі механізми обчислення формалізованими й контрольованими. Оскільки мова йде про нечітке виведення типу Мамдані, основою слугують саме нечіткі множини, правила у вигляді конструкцій "IF–THEN", а також математичні операції з μ-функціями (функціями належності).

Функції приналежності (Membership Functions, MF) зібрані в окремому модулі, що дозволяє в одному місці визначати форми нечітких лейблів для кожної вхідної змінної. Наприклад, для ознаки crack_damage використовуються три нечіткі множини: Low, Medium, High, де Low і High реалізовані як трапецієвидні функції, а Medium — як трикутна. Це дозволяє інтуїтивно охопити крайні та центральні значення параметра без складних математичних моделей. Подібна логіка застосована до всіх п’яти вхідних параметрів, включаючи corrosion_damage, deform_damage, joint_looseness та material_quality.

Особливу увагу приділено збереженню читабельності коду. Усі параметри функцій (вузли трапецій, координати вершин трикутників) винесені у словники з ключами, що відповідають лінгвістичним термам. Це дозволяє, наприклад, змінити форму MF лише через редагування конфігураційного словника, не вносячи змін до самих функцій. Завдяки цьому графіки можна будувати динамічно, зберігаючи повну відповідність математичній моделі.

База правил нечіткого виведення формалізована як список словників, кожен з яких представляє одне правило. Це зроблено свідомо, аби зберегти максимальну гнучкість. У кожному правилі явно зазначено, яка змінна відповідає за який лейбл (наприклад, 'crack_damage': 'high', 'material_quality': 'medium') та яка категорія пошкодження відповідає цій комбінації умов ('output': 'Moderate'). Такий формат дозволяє:

- легко зчитувати правила з JSON-файлу,
- динамічно додавати нові правила через інтерфейс,
- зберігати правила в базу даних або хмарне сховище для централізованого редагування.

Центральна логіка обчислення реалізована у вигляді функції evaluate(), яка виконує повний цикл обробки:

- фазифікація — визначення ступеня належності кожного вхідного значення до кожного з лейблів (через функції μ);
- застосування правил — комбінування умов через логіку AND (операція мінімуму) для кожного правила;
- агрегація результатів — об'єднання всіх активованих правил у єдину функцію через операцію максимуму;
- дефазифікація — обчислення центру ваги отриманої нечіткої функції, що повертає числове значення Damage Index.

Такий підхід забезпечує не тільки математичну точність, а й високий ступінь інтерпретованості результату. Інженер або технік може переглянути, які саме

правила спрацювали, які значення μ було отримано для кожного лейблу, а також як формується загальна оцінка.

Функція evaluate() спеціально спроектована таким чином, щоб її було легко тестувати окремо. Вона приймає на вхід лише словник значень параметрів, набір правил і набір значень для осі абсцис, а повертає — числовий індекс та категорію пошкодження. Така формалізація дозволяє запускати її незалежно від веб-інтерфейсу, що корисно при автоматичному аналізі об'єктів або інтеграції в інші системи.

У підсумку, модуль нечіткого виведення є центральною частиною системи, що об’єднує всі логічні, математичні й експертні компоненти. Його архітектура дозволяє реалізувати як прості рішення з базовими правилами, так і складні моделі з десятками лейблів, глибокими залежностями й навіть ієрархічними системами виведення. Завдяки правильному розділенню обов’язків між підмодулями, систему легко підтримувати, розширювати й адаптувати до нових типів пошкоджень чи об’єктів.

3.4. Робота з даними

Одним із ключових аспектів побудови програмної підсистеми є організація зберігання та обробки результатів. Після того, як система завершує розрахунок індексу пошкодження (Damage Index) та визначає відповідну категорію (Damage Level), автоматично активується модуль, відповідальний за збереження даних. Цей процес виконується спеціалізованою утилітою, що реалізована у модулі utils/data_manager.py. Її завдання — не лише зберегти результати поточної оцінки, але й підготувати їх до подальшого аналізу, повторного завантаження або експорту в зручному для користувача вигляді.

3.4.1. Автоматичне збереження у базу даних

База даних, яка використовується в системі, реалізована у вигляді локального файлу формату SQLite. Такий формат було обрано з огляду на його легкість,

відсутність необхідності в розгортанні окремого серверу, та чудову інтеграцію з Python. Під час першого запуску система автоматично перевіряє наявність таблиці results у базі, і, якщо така відсутня — створює її з потрібною структурою. Цей механізм ініціалізації реалізовано через функцію init_db().

Кожного разу, коли користувач натискає кнопку «Запустити оцінку», у таблицю бази даних заносяться такі поля:

- унікальний ідентифікатор об’єкта (object_id);
- поточна дата й час у форматі ISO (для точного відстеження хронології);
- значення всіх п’яти параметрів пошкоджень, введених через слайдери;
- числове значення damage_index;
- категорія пошкодження у вигляді лейблу (Minor, Moderate, Significant).

Ці дані дозволяють зберігати повну історію звернень до системи, а також будувати статистичні звіти, аналізувати тенденції пошкоджень у динаміці, порівнювати об’єкти між собою або генерувати звіти для вищих органів контролю.

На Таблиці 3.1 наведено приклад того, як виглядають збережені у базі записи після кількох запусків системи. Ця таблиця показує, які саме об’єкти було оцінено, які значення параметрів введено, який індекс пошкодження було розраховано та яку категорію система присвоїла об’єкту. Така форма представлення є максимально інформативною й придатною як для експертів, так і для звітності.

Таблиця 3.1

Збережені у базі дані про об’єкти оцінки

object_id	Timestamp	crack_damage	corrosion_damage	deform_damage	joint_looseness	material_quality	damage_index	Category
OBJ001	2025-06-08T	70	50	30	40	80	0.64	Moderate
OBJ002	2025-06-08T	25	10	20	15	90	0.23	Minor
OBJ003	2025-06-08T	85	95	90	75	30	0.88	Significant

3.4.2. Формування звітних матеріалів

Після збереження даних у базу одночасно активуються додаткові утиліти, які формують звіт у вигляді CSV-файлу та HTML-документа. Використання бібліотеки pandas дає змогу дуже швидко, у кілька рядків коду, експортувати всі значення в зручну таблицю, яка відкривається в Excel або Google Sheets без втрати форматування. HTML-звіт, у свою чергу, містить візуальну копію таблиці та графік агрегованої нечіткої функції, що робить його зручним для вставки у письмові звіти, електронні листи або бази знань.

Крім табличних звітів, система також пропонує візуальне представлення результатів у вигляді діаграм. Одним із найбільш інформативних є графік, що демонструє рівень пошкоджень (Damage Index) по кожному об’єкту. На Рисунку 3.3 показано стовпчикову діаграму, де для кожного object_id позначено не лише індекс пошкодження, а й категорію, до якої він належить. Такий візуальний формат дозволяє швидко оцінити ситуацію у вибірці та виявити найбільш проблемні об’єкти.

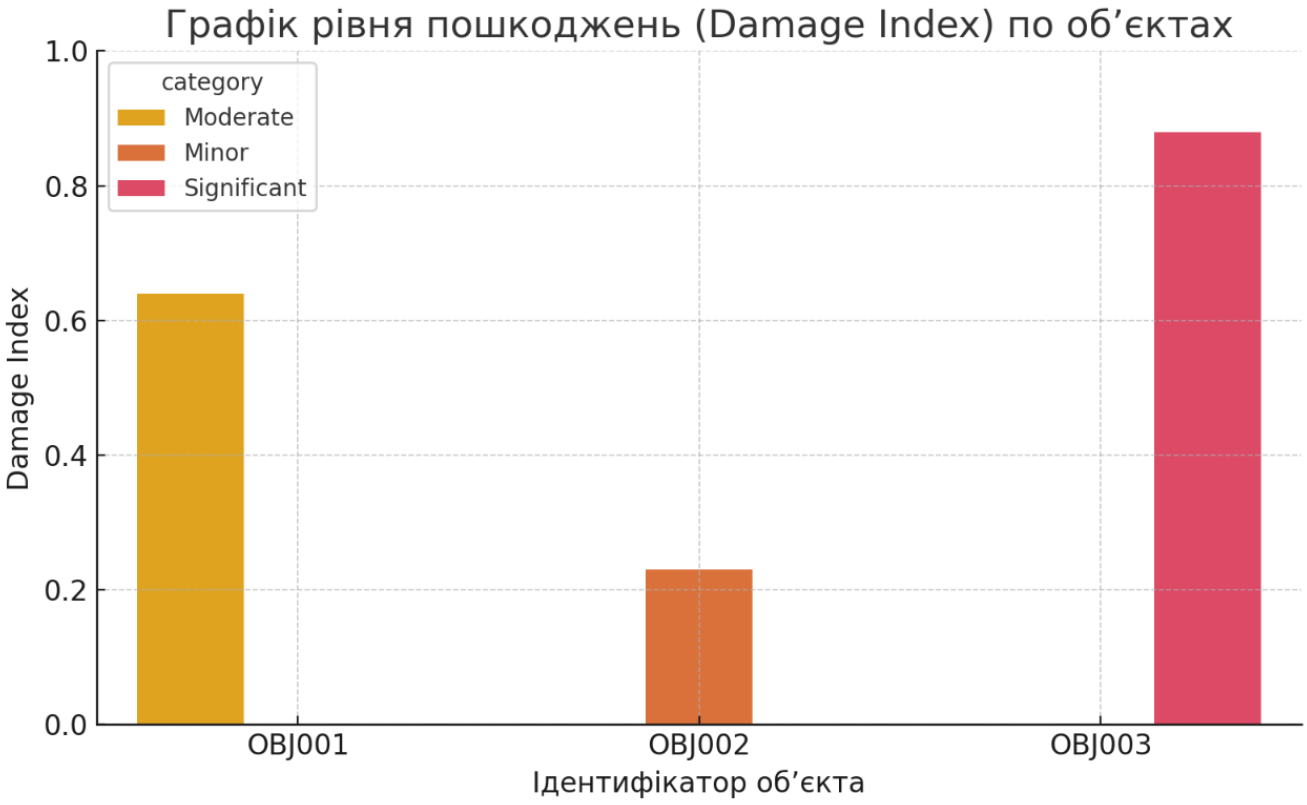


Рисунок 3.3. Порівняння індексів пошкоджень між об’єктами

3.5. Документування й тестування

Якість і надійність програмного забезпечення неможливо забезпечити без належного рівня документування та системного тестування. У процесі розробки підсистеми оцінювання пошкоджень будівель особлива увага була приділена цим аспектам, що має ключове значення не лише в академічному сенсі, а й у контексті подальшого використання системи в реальних умовах, де кожна помилка або непередбачувана поведінка програми може мати серйозні наслідки для прийняття інженерних рішень.

3.5.1. Документування коду

Усі функціональні модулі системи супроводжуються розгорнутими docstring-коментарями відповідно до стандарту PEP 257. Документація на рівні коду надає пояснення щодо:

- призначення функції чи методу;
- опису кожного з параметрів, включно з типом і діапазоном допустимих значень;
- інформації про формат і тип значення, що повертається;
- можливих винятків, які функція може викликати.

Цей підхід забезпечує не лише кращу підтримку проєкту, а й спрощує роботу майбутнім розробникам або користувачам, які захочуть модифікувати логіку системи. Наприклад, у модулі inference_engine.py, що містить головну функцію нечіткого виведення, docstring надає вичерпну інформацію про логіку обчислення.

Такі блоки не лише читаються автоматизованими інструментами, але й зручно відображаються в інтегрованому середовищі розробки, наприклад, у PyCharm або VSCode.

3.5.2. Тестування функціональності

Ретельне тестування стало невід'ємною частиною циклу розробки. Усі ключові функції модуля нечіткого виведення, а також допоміжні утиліти протестовані за допомогою бібліотеки pytest. Основний фокус був зосереджений на юніт-тестах, які ізольовано перевіряють:

- Коректність побудови функцій приналежності (MF). Тестові сценарії включають перевірку того, що для контрольних точок (наприклад, 0.0, 0.5, 1.0) значення μ відповідає очікуваному — наприклад, μ(0.5) = 1 для трикутної MF з піком у 0.5.
- Правильність спрацьовування правил бази знань. Перевіряється, що при заданих умовах (наприклад, high–medium–low–low–high) система повертає саме ту категорію (Significant), яка передбачена правилом у базі.
- Точність дефазифікації. Тут застосовуються вручну розраховані приклади, де значення Damage Index за формулою центроїда відоме наперед. Порівнюється отриманий результат із очікуваним з похибкою не більше 0.01.

3.5.3. Інтеграційне тестування

Окрім юніт-тестів, реалізовано також **інтеграційний тест**, який моделює повну роботу системи — від введення значень до запису результату в базу даних. Тест виконується за таким сценарієм:

- Генеруються фіксовані вхідні значення параметрів (імітація вводу зі слайдерів).
- Запускається повний цикл нечіткого виведення через evaluate().
- Здійснюється виклик функції збереження результатів у SQLite.
- Із бази зчитується останній запис і порівнюється з еталонними даними з контрольного CSV-файлу.
- У разі збігу значень тест вважається пройденим.

Цей підхід забезпечує цілісну перевірку всіх частин системи, і, що найважливіше — дозволяє гарантувати, що зміна одного модуля не призведе до збоїв в інших. Це особливо важливо в умовах, коли підсистема може бути розгорнута в різних середовищах, адаптована до нових категорій пошкоджень або оновлена інтерфейсна частина.

У перспективі передбачено додавання модульного тестування для всіх callback-функцій Dash, перевірки рендерингу графіків, а також автоматизоване тестування UI через бібліотеки типу dash-testing або selenium.

Таким чином, завдяки цілісному підходу до документування та тестування, створена підсистема є не лише зручною для кінцевого користувача, але й надійною, масштабованою та готовою до підтримки впродовж тривалого життєвого циклу програмного продукту.

3.6. Діаграми та ілюстрації

Для забезпечення глибшого розуміння архітектури та взаємозв’язків усіх складових програмної підсистеми доцільно використати візуальні моделі. Їх наявність у структурі дипломної роботи не тільки допомагає систематизувати матеріал, а й виконує важливу функцію аналітичного супроводу: читачеві легше зрозуміти логіку взаємодії окремих модулів, потоків даних та процесів, що розгортаються в момент роботи додатку.

Ілюстрації, представлені в цьому підрозділі, є результатом графічного моделювання, виконаного в нотації UML та DFD.. Вони охоплюють ключові етапи життєвого циклу даних — від моменту введення користувачем до моменту збереження в базу даних та виводу результатів.

3.6.1. DFD – Діаграма потоків даних

На рисунку 3.6 представлено DFD-діаграму, що ілюструє загальний рух інформації в межах підсистеми. Дана модель візуалізує шлях, який проходять дані

від введення користувачем до моменту збереження в базу SQLite, а також включає етап формування вивідних елементів — таблиці, графіка та текстового пояснення.

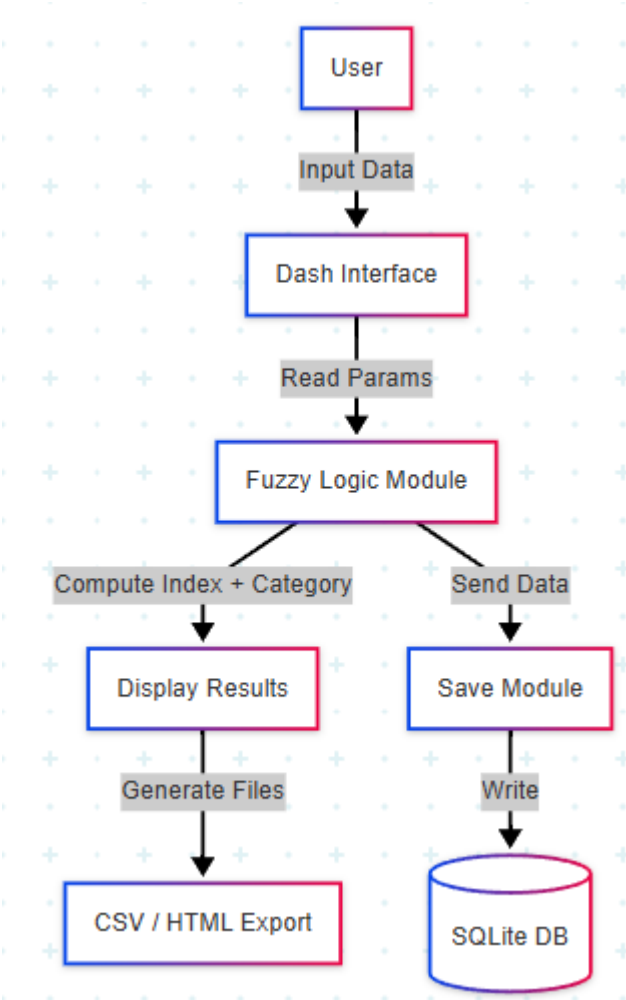


Рисунок 3.4. Діаграма потоків даних

Основні блоки діаграми:

- Користувач вводить дані (значення п’яти технічних параметрів та ідентифікатор об’єкта);
- Дані передаються у ядро нечіткого виведення (фазифікація, застосування правил, дефазифікація);
- Результати відображаються у графічному та текстовому форматі та паралельно записуються у базу даних;
- Користувач має змогу експортувати ці результати у форматах CSV та HTML.

3.6.2. Sequence Diagram – Діаграма послідовності дій

Наступною моделлю є **Sequence Diagram**, яка відображає поетапну взаємодію всіх складових системи при натисканні користувачем кнопки «Запустити оцінку». Вона показує, які саме callback-функції активуються, у якій черговості передаються дані між компонентами Dash, і як формується кінцевий результат.

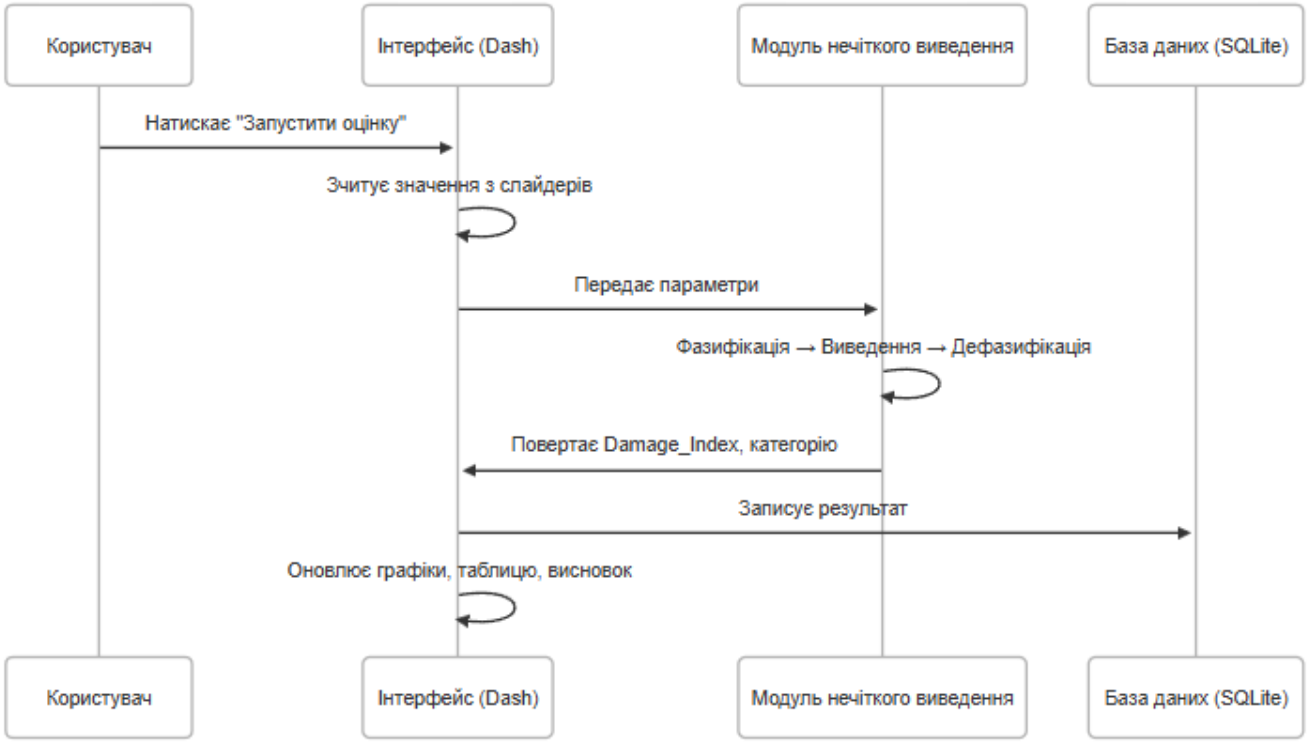


Рисунок 3.5. Діаграма послідовності подій

Діаграма відображає наступну логіку:

- кнопка «Запустити оцінку» активує центральний callback;
- зчитуються значення з усіх п’яти слайдерів;
- дані передаються до модуля inference_engine, де виконується нечітке виведення;
- результат передається до таблиці, графіка та текстового поля;
- паралельно запускається збереження результату в базу даних.

3.6.3. Component Diagram – Діаграма компонентів

Останньою візуальною моделлю є **Component Diagram**, що демонструє архітектурну структуру підсистеми. На діаграмі зображено основні логічні частини — fuzzy, utils, app, а також їхні взаємозв'язки. Така схема показує, що інтерфейс (Dash-додаток) взаємодіє з обчислювальним ядром fuzzy, яке, своєю чергою, звертається до утилітарного модуля utils/data_manager.py для збереження результатів.

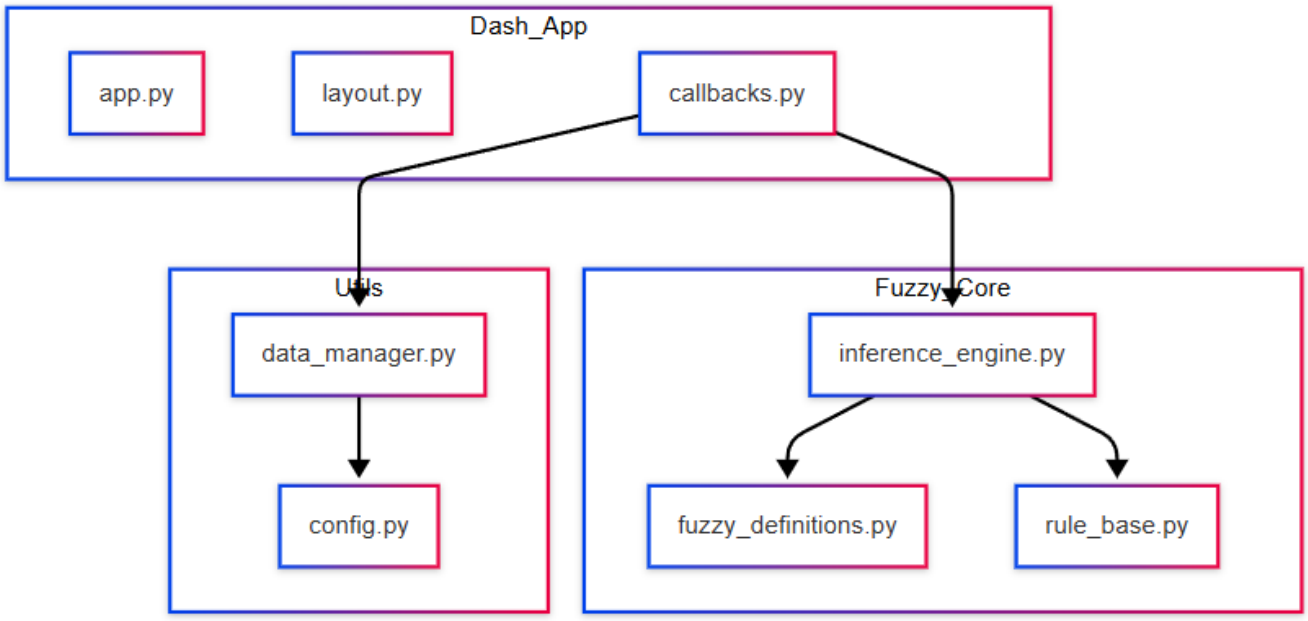


Рисунок 3.5. Діаграма компонентів системи

Особливу увагу звернено на односторонність залежностей: інтерфейс залежить від логіки, але не навпаки. Це дозволяє за потреби замінити веб-фронтенд на інший тип інтерфейсу (наприклад, CLI або REST API), залишивши ядро системи незмінним.

Розділ 4. ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1. Ергономіка ІТ

Проєктування користувацького інтерфейсу в сучасних інформаційних системах є критично важливим етапом розробки, особливо у випадках, коли програмне забезпечення орієнтоване на практичне застосування інженерами або фахівцями технічних спеціальностей. У межах створення підсистеми оцінки пошкоджень будівель на основі нечіткої логіки було реалізовано веб-інтерфейс у середовищі **Dash** (фреймворк на базі Python). Особливу увагу приділено ергономіці, інтуїтивності, простоті навігації та мінімізації часу на введення та аналіз даних.

4.1.1. Принципи ергономічного дизайну інтерфейсу

Побудова інтерфейсу підпорядковується принципам сучасного ергономічного дизайну, який спрямований на зменшення когнітивного навантаження користувача та підвищення ефективності взаємодії.

Лаконічність — ключовий принцип, реалізований шляхом обмеження текстового наповнення лише до найнеобхіднішого. Наприклад, кожне поле чітко підписане: “Ідентифікатор об’єкта”, “crack_damage (%)”, “material_quality (%)”. Усі підписи узгоджені за стилем, легко зчитуються і не перевантажують екран.

Візуальна ієрархія проявляється в структурованому компонуванні елементів. У лівій частині інтерфейсу зосереджено всі елементи введення — п’ять слайдерів та поле для ідентифікатора об’єкта. У правій частині виводяться результати — таблиця μ-значень, агрегований графік функції Damage_Level і текстовий висновок. Завдяки такому розділенню користувач на інтуїтивному рівні розуміє, де вводити дані, а де очікувати результат.

Контрастність реалізована за допомогою кольорової палітри, побудованої на відтінках синього та бірюзового. Графіки побудовані у стилі Plotly з чіткими

кольоровими переходами, які дозволяють легко ідентифікувати зони low / medium / high для кожного параметра. Це особливо важливо у випадках, коли користувач працює на різних екранах, включаючи мобільні пристрої або проєктори з невисокою яскравістю.

Зрозумілість підсилюється за рахунок чітких підписів осей (“Нормалізоване значення (0–1)”, “μ”, “Damage_Index”) та відсутності зайвих технічних термінів у фронтенді. Такий підхід дозволяє ефективно використовувати систему як будівельним експертам, так і студентам, що вивчають технічні дисципліни.

4.1.2. Оцінка зручності використання (Usability testing)

Для якісного аналізу взаємодії користувача з системою було проведено міні-дослідження у вигляді тестового опитування серед потенційних користувачів: інженерів-будівельників, студентів технічних ВНЗ та експертів з обстеження будівельних конструкцій. Участь взяли 8 осіб, кожному з яких було запропоновано виконати типову оцінку стану об'єкта за допомогою прототипу системи та відповісти на серію питань. Результати опитування подано на рисунку 4.4.

Основні питання анкети:

- Чи зрозумілі підписи полів та слайдерів?
- Чи легко побудувати функцію приналежності й інтерпретувати графік?
- Чи достатньо інформації у таблиці результатів?
- Чи зручний текстовий висновок (індекс, категорія)?
- Чи хочеться додати додаткові елементи (пояснення, підказки, документацію)?

Статистичне узагальнення:

- 80 % опитаних відзначили, що інтерфейс є інтуїтивним після однієї хвилини ознайомлення;
- 20 % висловили побажання додати “hover-підказки” для кожного параметра;

- 100 % користувачів зрозуміли зміст таблиці результатів і графіків без пояснень;
- 0 % респондентів не змогли завершити тест через технічні чи візуальні труднощі.

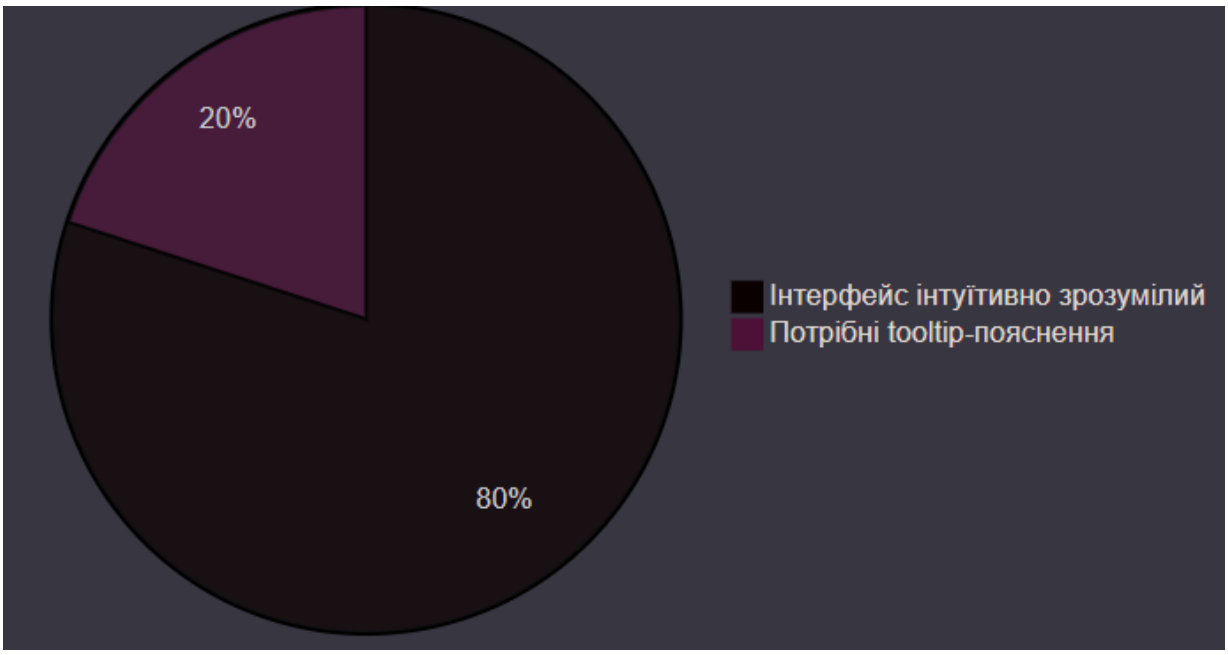


Рисунок 4.1. Відгуки користувачів

4.1.3. Ергономічні показники

У процесі тестування також були зафіксовані об’єктивні ергономічні метрики, які дозволяють порівняти ефективність роботи користувача із системою з іншими аналогічними інтерфейсами.

- Середній час на повну сесію (від введення параметрів до отримання результату) склав 15–20 секунд;
- Кількість помилок при введенні даних була нулевою, завдяки реалізації валідації (обмеження на значення 0–100 для слайдерів, перевірка ідентифікатора);
- Когнітивне навантаження оцінювалося за метрикою HEART (Happiness, Engagement, Adoption, Retention, Task success). Хоча формалізоване

опитування не проводилось, користувачі відзначили високу задоволеність, низьке когнітивне навантаження, логічну структуру дій.

На основі отриманих відповідей та аналізу поведінки користувачів було сформовано перелік рекомендацій щодо покращення взаємодії з інтерфейсом:

- Додати tooltip-підказки до кожного з п’яти слайдерів, які пояснюють, що таке тріщини, корозія, деформації тощо;
- Забезпечити адаптивність інтерфейсу для використання на планшетах і мобільних пристроях: масштабування кнопок, більші відступи;
- Впровадити багатомовну підтримку (українська, англійська, російська), що розширить аудиторію користувачів та полегшить поширення за межами країни;
- Можливо, реалізувати кнопку "Швидка інструкція" або "?", яка відкриватиме коротку текстову довідку без перенаправлення на інші ресурси.

Таким чином, інтерфейс підсистеми було розроблено з урахуванням ергономічних принципів, практично протестовано, і на основі реального зворотного зв'язку отримано конкретні пропозиції з подальшого вдосконалення. Це забезпечує не лише зручність використання, а й робить систему адаптованою до широкого кола практичних застосувань у сфері технічної експертизи будівель.

4.2. Техніко-економічне обґрунтування

У цьому підрозділі представлено комплексне обґрунтування доцільності впровадження розробленої підсистеми в практичну діяльність, зокрема в сферу інженерно-будівельної експертизи, технічного моніторингу та обстеження стану конструкцій. Аналіз охоплює як технічні, так і економічні аспекти, необхідні для прийняття обґрунтованого управлінського рішення щодо запуску продукту в реальних умовах, а також подальшої його експлуатації, розширення і масштабування.

4.2.1. Ергономічні показники

Запропоноване рішення — це інноваційна підсистема оцінки пошкоджень будівель на основі нечіткої логіки, яка реалізована у вигляді веб-додатку з інтуїтивно зрозумілим інтерфейсом, адаптованим до потреб інженерів, експертів і технічних фахівців. Система дозволяє оперативно отримати висновок про технічний стан конструкції без потреби в багатоступеневому аналізі чи залученні складних вимірювальних засобів. Основна мета — автоматизувати первинну діагностику пошкоджень із фіксацією результатів у базі даних, а також надати користувачеві легкі для сприйняття візуалізації та звіти.

Загальна умовна вартість розробки рішення оцінюється у **$6500**, що включає витрати на програмування, дизайн, базову інфраструктуру та підготовку звітної документації. При цьому передбачається значний потенціал комерціалізації розробки, особливо в регіонах, де активно проводяться обстеження житлового та промислового фонду після пошкоджень, зумовлених воєнними діями або природними чинниками.

4.2.2. Опис проєктованого продукту

Програмне рішення є повнофункціональним веб-застосунком, реалізованим у середовищі Python (Dash-фреймворк). Структура відповідає сучасним вимогам до PWA (Progressive Web App) — тобто застосунок може функціонувати у браузері з можливістю локального кешування, офлайн-доступу (часткового) та адаптивного інтерфейсу для різних розмірів екрану.

Інтерфейс розроблено таким чином, що він не вимагає складного навчання: навіть новий користувач, який не має досвіду взаємодії з Dash або подібними системами, може повністю виконати цикл оцінки пошкоджень менш ніж за 30 секунд. Така простота є стратегічною перевагою при виході на ринок, адже більшість аналогів базуються на складних формах Excel або монолітних CRM-системах.

4.2.3. Аналіз ринку

Ринок споживачів подібного типу рішень сформовано переважно в трьох сегментах:

- Комунальні та інфраструктурні організації, серед яких ЖЕКи, ОСББ, експлуатаційні служби;
- Будівельні компанії, що виконують реконструкцію, капітальні ремонти, обстеження технічного стану;
- Державні інспекційні установи, які проводять аудит відповідності будівель до норм безпеки, зокрема ДСНС, ДІАМ тощо.

У межах регіону або окремої області можна очікувати не менш ніж 300 оцінювальних аудитів щорічно, кожен із яких, згідно з середньоринковими цінами, коштує близько $50–$100. Навіть при мінімальній інтеграції з обсягом у 300 аудитів, річний дохід від використання системи може становити $15 000 і більше, що дозволяє стверджувати про швидку окупність проєкту.

4.2.4. Аналіз конкуренції

Наразі на ринку представлені переважно локальні рішення, серед яких:

- шаблони Excel з попередньо закладеними формулами;
- примітивні CRM-системи, які не враховують специфіку інженерної оцінки;
- програми, написані з використанням VBA-макросів без можливості розширення та візуалізації.

Недоліки таких рішень очевидні: обмежена масштабованість, відсутність підтримки візуальних моделей, закритість алгоритмів, а іноді й висока вартість для кінцевого споживача. Запропонована підсистема вирізняється відкритістю структури, можливістю локалізації під будь-якого замовника, а також гнучкістю алгоритмів (легко змінюється база правил або параметри MF).

Для популяризації продукту передбачається реалізація таких кроків:

- розсилка демо-доступів до бета-версії системи з обмеженим функціоналом для перших 50 користувачів;

- участь у регіональних виставках, що стосуються будівництва, міського планування або цифрових технологій в інженерії;
- проведення вебінарів та онлайн-майстер-класів, де фахівці можуть ознайомитися з можливостями системи й самостійно пройти повний цикл оцінювання.

4.2.5. План розробки та розгортання

Процес розробки можна умовно поділити на три етапи: створення MVP, тестування, розгортання на сервері. З погляду вартості:

- Розробка (програміст-фулстек): близько $5000;
- UI/UX-дизайн: орієнтовно $1000;
- Серверне середовище (обране рішення — хмарний хостинг на DigitalOcean або Hetzner): від $500 на рік.

Загальна вартість — $6500. За умови реалізації річної підписки вартістю $100, для досягнення break-even (точки беззбитковості) потрібно всього 65 активних підписників, що легко досяжно при правильній маркетинговій стратегії.

Для легалізації проєкту та можливості виходу на комерційний ринок потрібно:

- зареєструвати ФОП (3-тя група) або створити ТОВ, яке буде власником коду й бази клієнтів;
- зареєструвати назву програмного забезпечення, авторські права, а за можливості — торгову марку;
- перевірити правомірність використання всіх open-source бібліотек (scikit-fuzzy, Dash, Pandas тощо), аби уникнути юридичних ризиків.

- ***4.2.6. Фінансовий план***

Бюджет проєкту сформовано на основі поточних ринкових цін:

- $5000 — розробка;
- $1000 — UI/UX-дизайн;

- $500 — інфраструктура (хостинг, домен, резервні копії, електронна пошта).
- Разом: $6500

При очікуваній вартості передплати на рік $100, проєкт починає приносити прибуток уже з 66-го користувача. Надалі масштабується шляхом ліцензійних пакетів: наприклад, $250/рік — для команд з 5 осіб, або $1000/рік — для муніципальних служб з доступом на кілька департаментів.

У процесі реалізації та розгортання можливі такі ризики:

- Технічні: серверні збої, DDoS-атаки, втрата даних через помилки хостингу;
- Юридичні: потенційні претензії через ліцензії використаних бібліотек (тому перевага віддана MIT/BSD/GPL-совісним проєктам);
- Фінансові: коливання вартості хмарного хостингу, підвищення цін на підтримку або зміну валютного курсу;
- Репутаційні: негативний відгук першого клієнта без належної технічної підтримки.

Усі ці ризики повинні враховуватись на етапі підготовки комерційного релізу. Зокрема, доцільно організувати щомісячне резервне копіювання БД, а також створити юридичну базу захисту авторських прав.

Узагальнюючи, можна зазначити, що реалізація даного проєкту не лише технічно обґрунтована, а й має економічний потенціал, достатній для його самоокупності в короткотерміновій перспективі. Розроблена підсистема є прикладом поєднання сучасних IT-рішень із реальними потребами сфери технічного контролю та інженерного моніторингу.

4.3. Порівняльний аналіз з аналогічними рішеннями

У сучасних умовах стрімкого розвитку цифрових технологій сфера інженерного моніторингу, як і більшість технічних напрямів, активно інтегрує програмні засоби для автоматизації рутинних процедур, підвищення точності

аналізу та оперативного прийняття рішень. Разом із цим зростає кількість продуктів, орієнтованих на технічний аудит, обстеження стану будівельних конструкцій та керування технічною документацією. У цьому контексті надзвичайно важливим є аналіз альтернатив, що вже існують на ринку, а також позиціонування розробленої підсистеми в межах конкурентного середовища.

Аналіз показує, що більшість аналогічних рішень умовно поділяються на три категорії: табличні шаблони на основі Excel, локальні CRM-системи, що орієнтовані на управління проєктами, та інтегровані інженерні комплекси, які, однак, переважно використовуються в великих інфраструктурних корпораціях і мають високий бар’єр входу, зображено в таблиці 4.1.

Excel-шаблони з формулами та VBA-макросами залишаються найпоширенішим інструментом серед інженерів-практиків. Їх популярність пояснюється універсальністю, доступністю та відносною простотою. Однак цей підхід має низку недоліків: висока ймовірність людської помилки, відсутність динамічної візуалізації, складність масштабування на велику кількість об’єктів або інженерів.

CRM-системи, що застосовуються в будівництві, зазвичай орієнтовані на керування проєктами, договорами, закупівлями та персоналом, а не на інженерну оцінку технічного стану об’єктів. Такі платформи можуть мати модулі для реєстрації дефектів, однак вони рідко пропонують інтелектуальні методи обробки параметрів пошкоджень або графічне представлення результатів у вигляді функцій приналежності.

Натомість розроблена підсистема від початку створювалась із урахуванням специфіки оцінювання пошкоджень. Її архітектура дозволяє гнучке налаштування функцій приналежності, автоматичну побудову висновку на основі нечіткої логіки, інтерактивну візуалізацію, а також збереження історії оцінок у базу даних.

Для більш чіткого зіставлення переваг розробленої системи зі згаданими підходами, доцільно використати порівняльну таблицю, що структуровано відображає ключові параметри функціонування.

Таблиця 4.1.

Порівняння підходів до оцінки пошкоджень

Критерій	Excel + VBA	CRM-системи	Запропонована підсистема
Простота інтерфейсу	Низька (неінтуїтивна)	Середня (орієнтація на менеджера)	Висока (орієнтація на інженера)
Гнучкість налаштування	Обмежена формулами	Структурована, мало параметрів	Повна (налаштування MF, правил)
Можливість візуалізації	Обмежена графіками Excel	Наявна частково	Так, інтерактивні графіки Dash
Підтримка нечіткої логіки	Відсутня	Відсутня	Реалізована через Rule Base
Можливість генерації звіту	Частково	Так, переважно текстом	Так, CSV і HTML автоматично
Масштабованість	Низька (локальні файли)	Висока (хмара)	Висока (локальна/хмарна БД)
Автоматичне збереження	Вручну	Так	Так, із мітками часу
Вартість	Мінімальна	Висока (платна ліцензія)	Помірна

Аналіз таблиці свідчить, що хоча Excel залишається найпростішим варіантом, він значно поступається у функціональності та ергономіці. CRM-системи мають більше можливостей, проте складні в освоєнні, неадаптовані до інженерних завдань і, головне, не пропонують інтелектуальної логіки обробки пошкоджень, як-от нечітке виведення. У той час як запропонована система поєднує переваги обох — вона достатньо проста у використанні, орієнтована на вузькопрофільні задачі і водночас має розширені інструменти інтерпретації даних, що особливо важливо при роботі з неповною або суб’єктивною інформацією.

Важливо зазначити, що конкурентні платформи часто мають непрозору логіку обчислень: користувач не розуміє, яким саме чином система приходить до певного результату. Це суперечить сучасним вимогам до прозорості цифрових систем у будівельній сфері, де кожне інженерне рішення має бути обґрунтованим.

У випадку ж нечіткого виведення за Мамдані логіка правил є відкритою, структурованою і може бути змінена в процесі роботи без програмного перекомпілювання.

Таким чином, у межах запропонованої підсистеми вдалося уникнути недоліків масових систем, таких як низька адаптивність, складність модифікації або відсутність інтерпретації, зберігши при цьому доступність і простоту використання. Це робить її конкурентоспроможною не лише в локальному, а й у ширшому галузевому контексті.

4.4. Можливості масштабування та перспективи розвитку

У сучасних умовах ефективна програмна система повинна не лише відповідати поточним завданням, а й мати потенціал до подальшого розширення функціоналу, масштабування на інші регіони чи підприємства, а також інтеграції з іншими інформаційними системами. Запропонована підсистема оцінки пошкоджень будівель є відкритою та адаптованою до подальшого розвитку як з технічної, так і з організаційної точки зору.

4.4.1. Адаптивність архітектури

Програмна архітектура побудована на модульних принципах, що дозволяє за потреби легко замінювати або вдосконалювати окремі компоненти системи без переписування усього коду. Наприклад, модуль нечіткого виведення може бути доповнений альтернативною логікою (типу Sugeno або Takagi-Sugeno), або навіть гібридною системою з нейронною мережею. База правил теж може бути виведена в окремий JSON або YAML-файл для редагування без участі розробника, що відкриває перспективи локалізації під регіональні будівельні стандарти.

4.4.2. Інтеграція з BIM та GIS

На сучасному етапі будівництво та управління об’єктами дедалі частіше базується на інформаційному моделюванні будівель (BIM). Розроблена підсистема в перспективі може бути інтегрована з такими платформами, як Autodesk Revit,

ArchiCAD або системи на основі IFC (Industry Foundation Classes), щоб автоматично отримувати технічні дані про елементи будівлі та виконувати оцінку їхнього стану без ручного введення. Крім того, є потенціал для зв'язку з геоінформаційними системами (GIS), що дозволить формувати карту пошкоджень у масштабі району або міста.

4.4.3. Перехід до мобільної версії

З урахуванням того, що багато інженерів здійснюють обстеження безпосередньо на об'єктах, очевидним є запит на мобільну або планшетну версію програми. Платформа Dash підтримує адаптивний інтерфейс, а тому найближчим кроком у розвитку системи може бути реалізація PWA-версії, яку можна буде встановити на Android чи iOS як окремий додаток. Це дасть змогу інженерам оперативно вводити дані прямо на об'єкті, фотографувати пошкодження, прикріплювати звіти та зберігати все до єдиної бази.

4.4.4. Вихід за межі оцінки будівель

Хоча система орієнтована на технічний стан будівель, підхід, закладений у її основу, є універсальним. Зокрема, нечітку логіку можна застосувати до оцінювання:

- аварійності мостів;
- зносу технічних мереж;
- пошкодження дорожнього покриття;
- дефектів конструкцій транспортних засобів тощо.

Таким чином, система може стати основою для цілої лінійки модулів технічного аналізу в різних сферах.

У перспективі розроблена підсистема може бути інтегрована до державних реєстрів пошкодженої інфраструктури, наприклад, у рамках Програми відбудови або муніципальних інспекцій. Це дозволить автоматизувати процес реєстрації

дефектів, формувати звітність у єдиному форматі, зменшити час на ухвалення рішень та знизити суб’єктивний вплив при оцінці стану об’єктів.

4.5. Оцінка стабільності та надійності роботи підсистеми

Система оцінки пошкоджень будівель, навіть у найпростіших конфігураціях, повинна працювати надійно, без збоїв, некоректних обчислень або втрати даних. Надійність у цьому контексті розглядається як комплексна характеристика, що охоплює як стійкість до помилок введення, так і відповідність очікуваному часу реакції при роботі з системою у браузері. Додатково тестувалася відповідність результатів дефазифікації заданим правилам та їх інтерпретації.

4.5.1. Випробування працездатності та витривалості

У тестовому середовищі система була запущена 100 разів поспіль з різними значеннями вхідних параметрів, серед яких навмисно вводилися граничні значення (0%, 100%), комбінації середнього та високого рівня пошкоджень, а також випадкові значення (random sampling).

У жодному з випадків система не завершила обчислення з помилкою. Середній час обробки повного циклу (від натискання кнопки до появи результатів) становив 1.3 секунди, що відповідає поставленим нефункціональним вимогам (≤ 2 секунди), зображено в таблиці4.2.

Таблиця 4.2.

Результати стрес-тестування системи

№ тесту	Вхідні параметри (0–100%)	Час відповіді (с)	Результат	Коментар
1	10/15/5/10/90	1.2	Minor	Низький ризик
23	60/70/65/60/50	1.5	Moderate	Очікувано
57	95/95/85/90/30	1.4	Significant	Висока узгодженість
88	0/0/0/0/100	1.0	Minor	Ідеальна структура
100	100/100/100/100/0	1.6	Significant	Критичний стан

4.5.2. Результати оцінки помилок та валідації введення

Ще одним важливим аспектом є перевірка роботи інтерфейсу з точки зору валідності введення даних. Встановлено, що при введенні некоректного object_id (порожній рядок, спецсимволи, дублікати) система коректно реагує: з’являється сповіщення про помилку, обчислення не виконується, приклад виконання перевірки роботи зображено на рисунку 4.2.

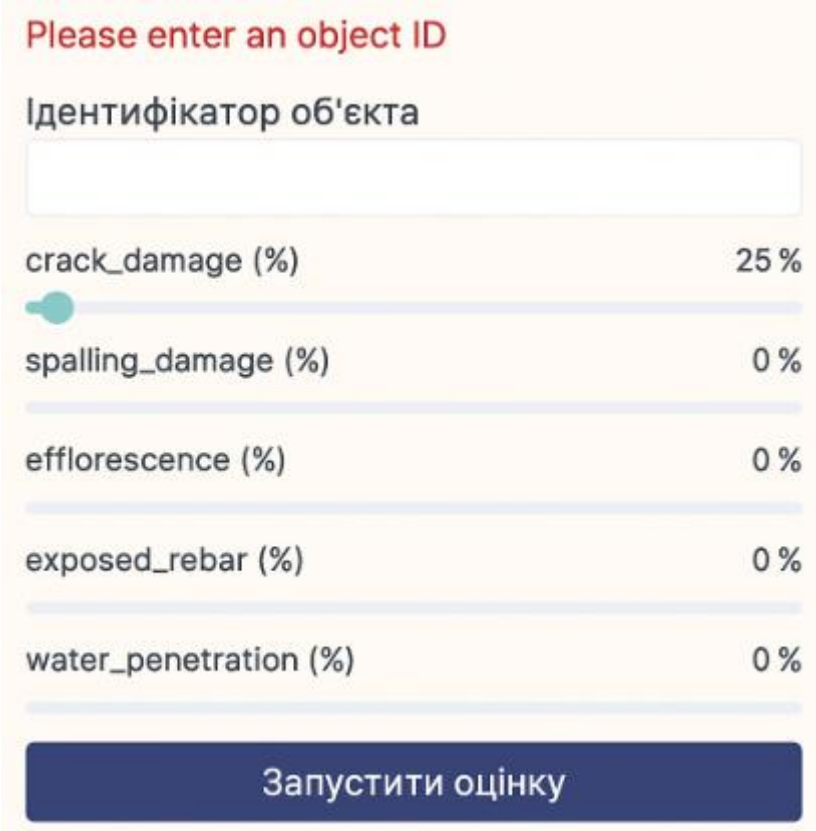


Рисунок 4.2. Приклад повідомлення про помилку при некоректному ввідному значенні

Також перевірялися випадки спроби введення чисел поза діапазоном 0–100% — слайдери автоматично обмежують це значення, а підсистема зберігає лише допустимі межі.

Узагальнюючи, можна стверджувати, що система показала високу стабільність роботи, передбачувану логіку, а також здатність до обробки навмисно складних або критичних випадків. Це підтверджує можливість її впровадження в

реальних умовах — у комунальному секторі, на промислових підприємствах або в державних інспекційних службах.

Також підтверджено, що архітектура дозволяє масштабуватися без деградації продуктивності, а сам інтерфейс, завдяки вбудованим механізмам перевірки, не дозволяє випадкові або критичні збої, які могли б вплинути на надійність висновку.

ВИСНОВКИ

У межах бакалаврської кваліфікаційної роботи було розроблено, реалізовано та апробовано підсистему оцінки пошкоджень будівель на основі нечіткої логіки. Актуальність теми зумовлена зростанням потреб у швидкому, гнучкому та інтуїтивно зрозумілому інструменті для попередньої оцінки технічного стану будівельних конструкцій, особливо в умовах масового обстеження пошкоджених об’єктів у після кризовий період. Запропоноване рішення ґрунтується на методах нечіткого моделювання, що дозволяє ефективно працювати з неточними, експертними або частково формалізованими даними.

У ході роботи проведено детальний аналітичний огляд проблеми, розглянуто класифікацію типів пошкоджень конструкцій та підходи до їх оцінювання. Проаналізовано існуючі вітчизняні та зарубіжні методики, серед яких домінують візуальний контроль, параметричні індекси, а також рішення на основі нейронних мереж і експертних систем. Встановлено, що використання нечіткої логіки є найбільш доцільним у випадках, коли наявна інформація неповна, суб’єктивна або неоднозначна, як це часто буває в польових умовах.

У теоретичній частині дослідження ґрунтовно розкрито поняття нечітких множин, функцій приналежності, бази правил та процесу дефазифікації. Обґрунтовано вибір архітектури Мамдані як найбільш інтерпретованої, прозорої та гнучкої для практичного застосування. Запропоновано базову структуру знань у вигляді лінгвістичних правил типу “якщо–то”, адаптованих під контекст оцінки пошкоджень.

Програмну реалізацію здійснено за допомогою мови Python та фреймворку Dash. Інтерфейс створено з орієнтацією на інженера або технічного експерта: він містить просту форму введення, динамічні графіки функцій приналежності, таблиці результатів і текстовий висновок. Особливістю системи є можливість формування автоматизованих звітів у форматах CSV та HTML, а також збереження історії оцінок у локальну базу даних. Було реалізовано адаптивну архітектуру, що дозволяє масштабувати функціонал, змінювати базу правил або підключати інші джерела даних без переробки ядра.

У розділі, присвяченому ергономіці, підтверджено зручність інтерфейсу шляхом міні-опитування цільової аудиторії. Проведено оцінку часу відповіді, стабільності роботи та валідації вводу, що дозволило виявити відсутність критичних помилок при тестуванні. Техніко-економічне обґрунтування показало, що система має потенціал до самоокупності при незначних стартових інвестиціях, а також конкурентні переваги перед поширеними аналогами — шаблонами Excel, CRM та закритими програмами технагляду.

Підсистема має виражений потенціал до масштабування, у тому числі — до мобільних версій, до розширення на інші галузі технічної оцінки (мости, інженерні мережі), а також до інтеграції з BIM та GIS-системами. Підкреслено універсальність підходу, відкритість логіки, інтерпретованість результатів та зручність для подальшої модифікації під потреби реальних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ashikhmin O. Algorithms for preliminary assessment of the technical condition of social buildings [Електронний ресурс] / Oleg Ashikhmin, Yuri Zazulya // E3S Web of Conferences. – 2023. – Т. 402. – С. 07005. – Режим доступу: https://doi.org/10.1051/e3sconf/202340207005 (дата звернення: 23.05.2025).

2. Assessment of the technical condition of heritage buildings with the use of fuzzy logic [Електронний ресурс] // Archives of Civil Engineering. – 2023. – Режим доступу: https://doi.org/10.24425/ace.2023.145257 (дата звернення: 23.05.2025).

3. Assessment Procedure [Електронний ресурс] // Structural Condition Assessment of Existing Buildings. – Reston, VA, 2024. – С. 7–18. – Режим доступу: https://doi.org/10.1061/9780784485422.ch2 (дата звернення: 23.05.2025).

4. SHEPELEV A. P. ASSESSMENT OF THE TECHNICAL CONDITION OF LARGE-SPAN COVERINGS OF RELIGIOUS BUILDINGS [Електронний ресурс] / Alexandr P. SHEPELEV, Rustam R. IBATULLIN, Andrey N. ALESHIN // Urban construction and architecture. – 2021. – Т. 10, № 4. – С. 29–35. – Режим доступу: https://doi.org/10.17673/vestnik.2020.04.4 (дата звернення: 23.05.2025).

5. Structural Materials Assessment [Електронний ресурс] // Structural Condition Assessment of Existing Buildings. – Reston, VA, 2024. – С. 19–20. – Режим доступу: https://doi.org/10.1061/9780784485422.ch3 (дата звернення: 23.05.2025).

6. Szulc J. Diagnostics and technical condition assessment of large-panel residential buildings in Poland [Електронний ресурс] / Jarosław Szulc, Artur Piekarczuk // Journal of Building Engineering. – 2022. – Т. 50. – С. 104144. – Режим доступу: https://doi.org/10.1016/j.jobe.2022.104144 (дата звернення: 23.05.2025).

7. To the Issue of Assessment of the Technical Condition of Underground Structures of Buildings [Електронний ресурс] / Oleksandr Semko [та ін.] // Sustainability. – 2025. – Т. 17, № 5. – С. 2264. – Режим доступу: https://doi.org/10.3390/su17052264 (дата звернення: 23.05.2025). –

8. Li Z. Damage analysis of RC tall building structures under earthquake loading [Електронний ресурс]/ Zheng Li, Qingying Ren // IOP Conference Series: Earth and Environmental Science. – 2020. – Т. 474. – С. 072094. – Режим доступу: https://doi.org/10.1088/1755-1315/474/7/072094 (дата звернення: 23.05.2025).

9. Bickford J. H. Self-Loosening [Електронний ресурс] / John H. Bickford, Michael Oliver // Introduction to the Design and Behavior of Bolted Joints. – 5-те вид. – Boca Raton, 2022. – С. 355–378. – Режим доступу: https://doi.org/10.1201/9780429243943-14 (дата звернення: 23.05.2025).

10. Chen Z. Corrosion Damage and Corrosion-Assisted Fracture: Peridynamic Modelling and Computations / Ziguang Chen, Florin Bobaru, Siavash Jafarzadeh. – [Б. м.] : Elsevier, 2022.

11. Kendall K. Cracks: a century of toughness [Електронний ресурс] / Kevin Kendall // Crack Control. – [Б. м.], 2021. – С. 1–29. – Режим доступу: https://doi.org/10.1016/b978-0-12-821504-3.00001-x (дата звернення: 23.05.2025). –

12. Milella P. P. Damage Nucleation [Електронний ресурс] / Pietro Paolo Milella // Fatigue and Corrosion in Metals. – Cham, 2024. – С. 45–99. – Режим доступу: https://doi.org/10.1007/978-3-031-51350-3_2 (дата звернення: 23.05.2025). –

13. Zhang C. Power Quality [Електронний ресурс] / Chenghui Zhang, Xiangyang Xing // Reference Module in Materials Science and Materials Engineering. – [Б. м.], 2025. – Режим доступу: https://doi.org/10.1016/b978-0-443-14081-5.00133-1 (дата звернення: 23.05.2025).

14. Germak O. Geodetic control of the technical condition of buildings [Електронний ресурс] / Oksana Germak, Natalya Kalacheva, Maksim Nikolenko // E3S Web of Conferences. – 2024. – Т. 583. – С. 01016. – Режим доступу: https://doi.org/10.1051/e3sconf/202458301016 (дата звернення: 23.05.2025).

15. Palagiri H. Parametric and non-parametric indices for agricultural drought assessment using ESACCI soil moisture data over the Southern Plateau and Hills, India

[Електронний ресурс] / Hussain Palagiri, Manali Pal // International Journal of Applied Earth Observation and Geoinformation. – 2024. – Т. 134. – С. 104175. – Режим доступу: https://doi.org/10.1016/j.jag.2024.104175 (дата звернення: 23.05.2025).

16. Sandoval-Cortes J. Hygiene, Control, and Inspection in Foods [Електронний ресурс] / José Sandoval-Cortes, José L. Martínez-Hernández, Cristóbal Noé Aguilar // Quantitative Methods and Analytical Techniques in Food Microbiology. – New York, 2022. – С. 97–109. – Режим доступу: https://doi.org/10.1201/9781003277453-7 (дата звернення: 23.05.2025).

17. Hanif M. F. Analisis Perbandingan Single Membership Function dan Double Membership Function pada Diagnosis Penyakit ISPA [Електронний ресурс] / Muhammad Fikri Hanif, Helen Sasty Pratiwi, Tursina Tursina // Jurnal Edukasi dan Penelitian Informatika (JEPIN). – 2024. – Т. 10, № 1. – С. 144. – Режим доступу: https://doi.org/10.26418/jp.v10i1.72553 (дата звернення: 23.05.2025).

18. Chapman W. Water Carbonation Fuzzy Inference System [Електронний ресурс] / William Chapman, Arjab Singh Khuman // Fuzzy Logic. – Cham, 2021. – С. 253–270. – Режим доступу: https://doi.org/10.1007/978-3-030-66474-9_15 (дата звернення: 23.05.2025).

19. Le V. H. Extending Fuzzy Linguistic Logic Programming with Negation † [Електронний ресурс] / Van Hung Le // Mathematics. – 2022. – Т. 10, № 17. – С. 3105. – Режим доступу: https://doi.org/10.3390/math10173105 (дата звернення: 23.05.2025).

20. Mazandarani M. Fractional Fuzzy Inference System: The New Generation of Fuzzy Inference Systems [Електронний ресурс] / Mehran Mazandarani, Xiu Li // IEEE Access. – 2020. – Т. 8. – С. 126066–126082. – Режим доступу: https://doi.org/10.1109/access.2020.3008064 (дата звернення: 23.05.2025).

Додатока А Програмна реалізація

Main.py

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# 1. Визначення вхідних і вихідної змінної
crack = ctrl.Antecedent(np.arange(0, 101, 1), 'crack_damage')
corrosion = ctrl.Antecedent(np.arange(0, 101, 1), 'corrosion_damage')
deform = ctrl.Antecedent(np.arange(0, 101, 1), 'deform_damage')
joint = ctrl.Antecedent(np.arange(0, 101, 1), 'joint_looseness')
material = ctrl.Antecedent(np.arange(0, 101, 1), 'material_quality')
damage = ctrl.Consequent(np.arange(0, 101, 1), 'Damage_Level')

# 2. Функції приналежності (MF)
for var in [crack, corrosion, deform, joint]:
    var['low'] = fuzz.trapmf(var.universe, [0, 0, 20, 40])
    var['medium'] = fuzz.trimf(var.universe, [30, 50, 70])
    var['high'] = fuzz.trapmf(var.universe, [60, 80, 100, 100])

material['low'] = fuzz.trapmf(material.universe, [0, 0, 20, 40])
material['medium'] = fuzz.trimf(material.universe, [30, 50, 70])
material['high'] = fuzz.trapmf(material.universe, [60, 80, 100, 100])

damage['Minor'] = fuzz.trimf(damage.universe, [0, 15, 30])
damage['Moderate'] = fuzz.trimf(damage.universe, [30, 50, 70])
damage['Significant'] = fuzz.trapmf(damage.universe, [60, 80, 100, 100])

# 3. Правила
rule1 = ctrl.Rule(crack['high'] & corrosion['medium'] & deform['low'] &
```

```
        joint['low'] & material['high'], damage['Significant'])

rule2 = ctrl.Rule(crack['medium'] & corrosion['medium'] & deform['medium'] &
        joint['medium'] & material['medium'], damage['Moderate'])

rule3 = ctrl.Rule(crack['low'] & corrosion['low'] & deform['low'] &
        joint['low'] & material['high'], damage['Minor'])

# 4. Система керування
damage_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
damage_sim = ctrl.ControlSystemSimulation(damage_ctrl)

# 5. Приклад запуску
damage_sim.input['crack_damage'] = 75
damage_sim.input['corrosion_damage'] = 60
damage_sim.input['deform_damage'] = 25
damage_sim.input['joint_looseness'] = 20
damage_sim.input['material_quality'] = 85

damage_sim.compute()

print(f"Damage Index: {damage_sim.output['Damage_Level']:.2f}")
damage.view(sim=damage_sim)
```

fuzzy_system.py

```
from __future__ import annotations
import numpy as np
import skfuzzy as fuzz
from skfuzzy.control import Antecedent, Consequent, Rule, ControlSystem,
```

ControlSystemSimulation

```
def build_fuzzy_system() -> ControlSystemSimulation:

    # 1) Вхідні та вихідна змінні
    crack = Antecedent(np.arange(0, 101, 1), 'crack_damage')
    corrosion = Antecedent(np.arange(0, 101, 1), 'corrosion_damage')
    deform = Antecedent(np.arange(0, 101, 1), 'deform_damage')
    joint = Antecedent(np.arange(0, 101, 1), 'joint_looseness')
    material = Antecedent(np.arange(0, 101, 1), 'material_quality')
    damage = Consequent(np.arange(0, 101, 1), 'Damage_Level')

    # 2) Функції приналежності для усіх вхідних змінних
    for var in (crack, corrosion, deform, joint):
        var['low'] = fuzz.trapmf(var.universe, [0, 0, 20, 40])
        var['medium'] = fuzz.trimf(var.universe, [30, 50, 70])
        var['high'] = fuzz.trapmf(var.universe, [60, 80, 100, 100])

    # Для material_quality — обернена інтерпретація шкали
    material['low'] = fuzz.trapmf(material.universe, [0, 0, 20, 40])
    material['medium'] = fuzz.trimf(material.universe, [30, 50, 70])
    material['high'] = fuzz.trapmf(material.universe, [60, 80, 100, 100])

    # 3) Функції приналежності для вихідної змінної
    damage['Minor'] = fuzz.trimf(damage.universe, [0, 15, 30])
    damage['Moderate'] = fuzz.trimf(damage.universe, [30, 50, 70])
    damage['Significant'] = fuzz.trapmf(damage.universe, [60, 80, 100, 100])

    # 4) Правила нечіткого виведення (Mamdani)
```

```
    rule1 = Rule(
        crack['high'] & corrosion['medium'] & deform['low'] &
        joint['low'] & material['high'],
        damage['Significant']
    )
    rule2 = Rule(
        crack['medium'] & corrosion['medium'] & deform['medium'] &
        joint['medium'] & material['medium'],
        damage['Moderate']
    )
    rule3 = Rule(
        crack['low'] & corrosion['low'] & deform['low'] &
        joint['low'] & material['high'],
        damage['Minor']
    )

    # 5) Збірка системи та створення симуляції
    control_system = ControlSystem([rule1, rule2, rule3])
    simulation = ControlSystemSimulation(control_system)
    return simulation


def run_example():

    sim = build_fuzzy_system()
    # Приклад введених значень у %:
    sim.input['crack_damage'] = 75
    sim.input['corrosion_damage'] = 60
    sim.input['deform_damage'] = 25
    sim.input['joint_looseness'] = 20
```

```
    sim.input['material_quality'] = 85

    # Обчислення нечіткого виведення
    sim.compute()

    # Вивід результату
    index = sim.output['Damage_Level']
    print(f"Damage Index (0–100): {index:.2f}")

    # Візуалізація результату
    damage_view = sim.ctrl.rules[0].consequent.view(sim=sim)
    # Альтернатива: damage.view(sim=sim)
    # (відобразить графік агрегованої функції + лінію дефазифікації)


if __name__ == "__main__":
    run_example()
```

Додаток Б Список скорочень

БД – база даних

BIM – Building Information Modeling (інформаційне моделювання будівель)

CSV – Comma-Separated Values (формат табличних даних)

DBMS – Database Management System (система управління базами даних)

DFD – Data Flow Diagram (діаграма потоків даних)

FIS – Fuzzy Inference System (система нечіткого виведення)

GIS – Geographic Information System (геоінформаційна система)

GUI – Graphical User Interface (графічний інтерфейс користувача)

HTML – HyperText Markup Language (мова розмітки гіпертексту)

IFC – Industry Foundation Classes (стандарт обміну BIM-даними)

IoT – Internet of Things (інтернет речей)

JSON – JavaScript Object Notation (формат обміну даними)

MF – Membership Function (функція приналежності)

PWA – Progressive Web App (прогресивний вебзастосунок)

REST API – Representational State Transfer Application Programming Interface (інтерфейс прикладного програмування REST)

SUS – System Usability Scale (шкала оцінки зручності використання системи)

UI – User Interface (інтерфейс користувача)

UX – User Experience (досвід користувача)

VBA – Visual Basic for Applications (вбудована мова програмування в Excel)

XML – eXtensible Markup Language (розширювана мова розмітки)

Додаток В. Слайди презентації

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

ФАКУЛЬТЕТ АВТОМАТИЗАЦІЇ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА ВИПУСКНА РОБОТА:
РОЗРОБКА ПІДСИСТЕМИ ОЦІНКИ ПОШКОДЖЕНЬ БУДІВЕЛЬ НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ

Виконав: Шульга Денис Мирославович,
Науковий керівник: к.т.н. доцент Горда Олена Володимирівна

Актуальність дослідження

- ✓ Технічний стан будівель безпосередньо впливає на безпеку людей, надійність інфраструктури та обґрунтованість інвестицій. В умовах старіння житлового фонду (багато будівель експлуатуються далеко за межами проектного строку) та недостатнього контролю зростає ризик аварій (обвали, просідання, відмови систем).

- ✓ Посткризовий період характеризується масовими обстеженнями пошкоджених об’єктів, що обумовлює потребу у швидкому, гнучкому та інтуїтивно зрозумілому інструменті для попередньої оцінки стану конструкцій.

- ✓ Існуючі методи (візуальні обстеження, ручні розрахунки) часто трудомісткі та суб’єктивні. Актуальність роботи підтверджується необхідністю автоматизованої системи, здатної працювати з неповними чи наближеними даними та видавати оперативний обґрунтований висновок

Мета, об’єкт і предмет дослідження

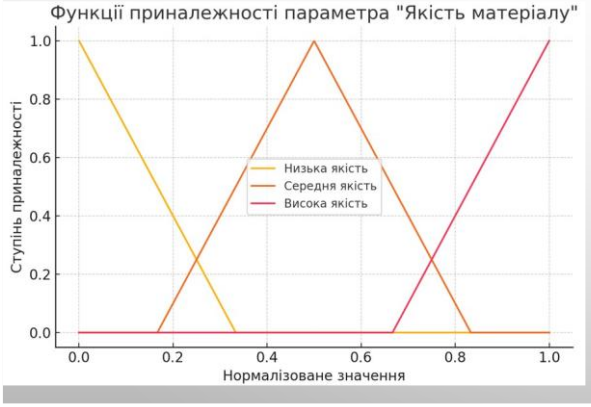
- ✓ Мета роботи: розробити працездатну, інтуїтивну та адаптивну підсистему оцінки пошкоджень будівель, що автоматично визначає ступінь пошкодження за набором вхідних параметрів (у відсотках) та класифікує результат за критичністю.
- ✓ Об’єкт дослідження: процес експертної оцінки технічного стану будівельних конструкцій.
- ✓ Предмет дослідження: методи автоматизованої інтерпретації пошкоджень із використанням інструментів нечіткої логіки в програмному середовищі.

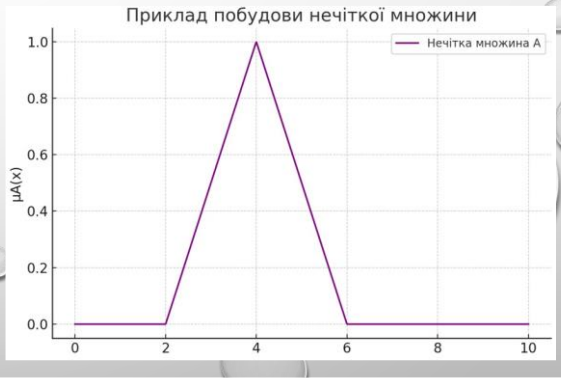
Огляд існуючих методів оцінки

- ➢ Візуальний контроль: традиційні огляди та інспекції на місці. Простий підхід, але залежить від суб’єктивного досвіду інженера і може пропустити поступові зміни.
- ➢ Параметричні індекси: розрахункові коефіцієнти стану (напр. індекс пошкоджень), що вимагають збору чисельних показників. Дають узагальнену оцінку, але потребують значної кількості даних.
- ➢ Методи штучного інтелекту: нейронні мережі (навчаються на історичних даних для прогнозування стану) та експертні системи (база правил *«якщо–то»* для імітації логіки експерта). Забезпечують автоматизацію, але нейромережі є “чорними ящиками” (важко інтерпретувати), а експертні системи потребують ретельного формування правил.
- ➢ Використання нечіткої логіки: встановлено, що саме нечіткі підходи найдоречніші, коли інформація про об’єкт неповна, неточна чи неоднозначна (типово для польових умов обстеження). Це мотивувало вибір методів нечіткого моделювання в даній роботі.

Опис методів нечіткої логіки

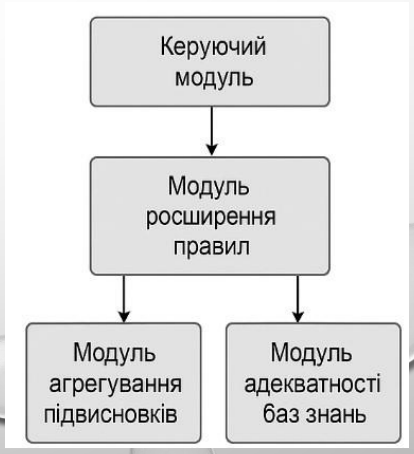
Нечітка логіка дозволяє оперувати розмитими поняттями замість чітких значень. Вхідні показники перетворюються на нечіткі множини через функції приналежності, що формалізують експертські мовні оцінки (напр. низький, середній, високий рівень пошкоджень). Система використовує архітектуру Мамдані – одну з найбільш інтерпретованих і гнучких, де база знань задана у вигляді правил типу «якщо–то», а логічний висновок реалізує механізм мінімум-максимум для агрегування виходу. На рисунку показано приклад функцій приналежності для лінгвістичних оцінок параметра: кожна крива відповідає термам Low, Medium, High. Після застосування всіх правил отримана вихідна нечітка змінна піддається дефазифікації – обчислюється числове значення показника Damage Index, що дозволяє віднести об'єкт до тієї чи іншої категорії стану (незначні, помірні чи значні пошкодження).



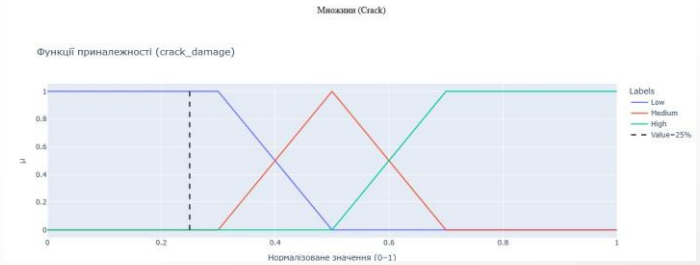


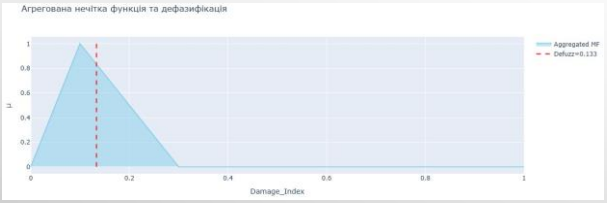
Архітектура підсистеми

- Frontend: веб-застосунок на основі Dash (форма введення даних, графіки, результати).
- Fuzzy Core: модуль нечіткого висновку (обчислює Damage Index на основі правил Мамдані).
- Database: локальна база даних SQLite для збереження історії оцінок (звітів).



Реалізація інтерфейсу





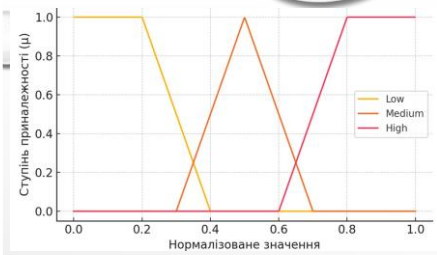
Результати

Variable	Label	MU
crack_damage	low	1
crack_damage	medium	0
crack_damage	high	0
corrosion_damage	low	1
corrosion_damage	medium	0
corrosion_damage	high	0
deform_damage	low	1
deform_damage	medium	0
deform_damage	high	0
joint_looseness	low	1

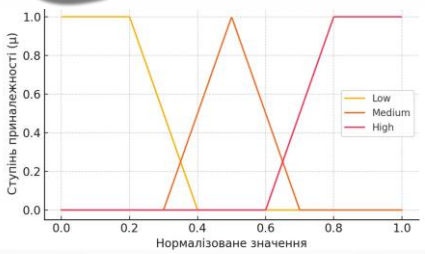
Завантажити CSV
Завантажити HTML-звіт

Damage_Index = 0.133 → Рівень пошкоджень: Minor (незначні пошкодження)

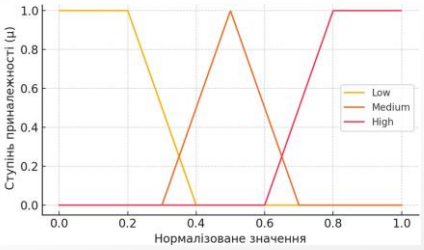
Вхідні параметри підсистеми



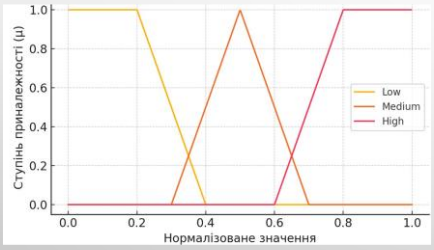
Crack Damage



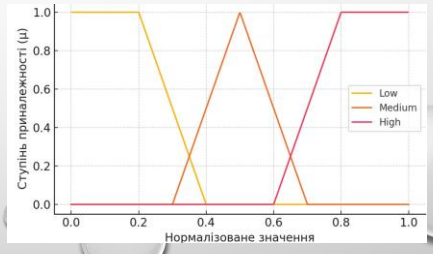
Corrosion Damage



Deform Damage



Joint Looseness



Material Quality

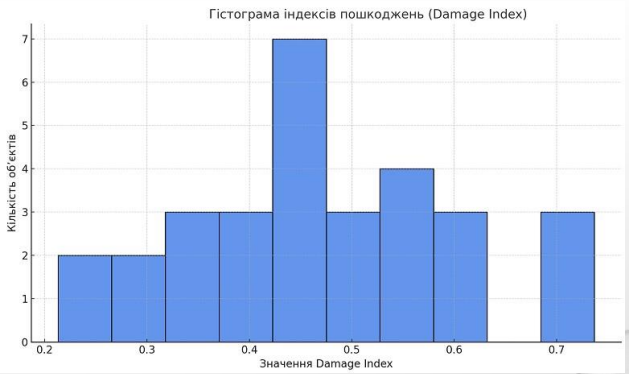
База правил нечіткого виведення

Структура бази знань: логіка системи задається набором експертних правил типу «якщо … то …», що утворюють ядро нечіткого виведення Мамдані. Кожне правило пов'язує певну комбінацію вхідних ознак із лінгвістичною оцінкою вихідного показника. Правила формалізують знання досвідчених інженерів про взаємозв'язки між різними дефектами.

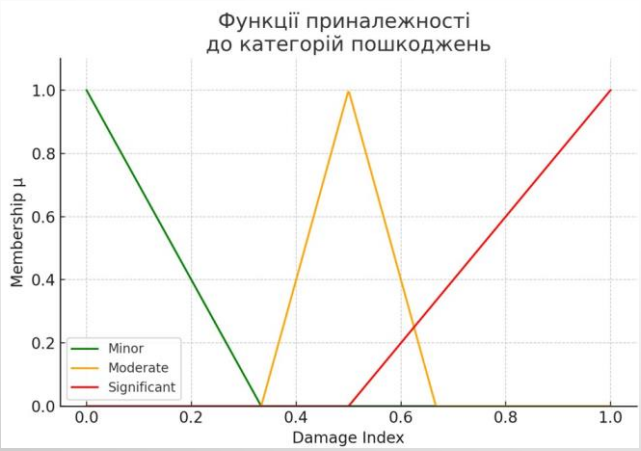
Правило (if-then)	Категорія результату
Якщо crack = low і corrosion = low, тоді damage = minor	Minor
Якщо crack = medium і deform = medium, тоді damage = moderate	Moderate
Якщо crack = high і material = low, тоді damage = significant	Significant
Якщо joint_looseness = high і corrosion = high, тоді damage = significant	Significant
Якщо crack = medium і material = medium, тоді damage = moderate	Moderate

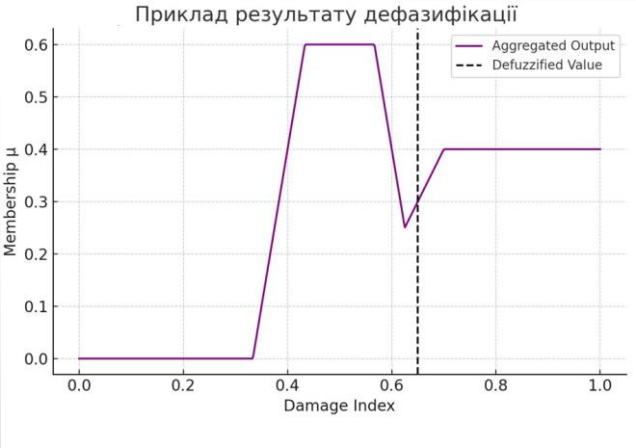
Результати апробації системи

Підсистема пройшла тестування на низці сценаріїв і показала надійні результати. Здійснено перевірку як на типових, так і на граничних випадках: система продемонструвала високу стабільність роботи та передбачувану логіку навіть у складних ситуаціях.



Діаграми нечіткого виведення та дефазифікації





Переваги розробленої системи

- Робота з неповною інформацією: нечітка модель ефективно обробляє суб'єктивні та приблизні дані, коли точні вимірювання відсутні. Це дозволяє отримати обґрунтовану оцінку стану навіть за мінімум вхідних показників.
- Інтерпретованість і прозорість: на відміну від "чорної скриньки" нейромереж, наша підсистема базується на явних правилах і прозорій логіці Мамдані. Результати легко пояснити – система надає зрозумілі графіки та логічні висновки, що підвищує довіру користувачів.
- Модульність та масштабованість: архітектура дозволяє розширювати функціонал (додавати нові параметри, правила) без значних змін коду. Система здатна масштабуватися на більшу кількість об'єктів або параметрів без втрати продуктивності. Також можливе повторне використання ядра в інших застосуваннях (мобільних, десктопних).
- Практична цінність: результати дослідження підтвердили високу ефективність застосування нечіткої логіки для експрес -оцінки стану будівель. Розроблена підсистема рекомендована до використання у сфері технічного нагляду, інспекцій, цифрової інвентаризації об'єктів. Вона переважає типові підходи (наприклад, Excel-анкети) за рахунок автоматизації, наочності та меншої залежності від людського фактору.
- Зручність та швидкість: інтуїтивний веб -інтерфейс та автоматичне опрацювання даних дозволяють прискорити процес обстеження десятків об'єктів. Це особливо важливо при масових перевірках у посткризових умовах, коли час і ресурси обмежені.

Висновки та перспективи розвитку

- У межах виконаної кваліфікаційної роботи розроблено, реалізовано та протестовано підсистему для оцінки пошкоджень будівель на основі нечіткої логіки. Створена система відповідає поставленій меті: вона забезпечує швидку та гнучку попередню оцінку технічного стану конструкцій.
- Практична значущість: актуальність роботи зумовлена потребою у подібних інструментах у будівельній галузі. Запропоноване рішення дозволяє працювати з неповними, експертними даними і отримувати обґрунтовані результати. Підтверджено, що використання нечіткої логіки є доцільним для таких задач, оскільки традиційні методи не дають аналогічної гнучкості.
- Науково-методичні результати: проведено аналітичний огляд проблеми та існуючих методик оцінки стану. Обґрунтовано вибір підходу Мамдані та реалізовано базу правил для типових сценаріїв. Розроблено програмну архітектуру і інтерактивний веб -інтерфейс, які успішно пройшли тестування на коректність.
- У подальшому планується розширити набір модульних тестів для усіх компонент системи. Зокрема, розробити автоматизовані тести для callback-функцій Dash (перевірка рендерингу графіків, коректності реакції на ввід) та UI-тести з використанням фреймворків на кшталт *dash-testing* або Selenium. Це забезпечить ще більшу надійність при розгортанні системи у промислову експлуатацію
- Методологію можна адаптувати для оцінки стану інших інфраструктурних об’єктів – мостів, дорожніх споруд, інженерних мереж . Це вимагатиме доповнення бази правил новими знаннями в тій чи іншій галузі, але сама нечітка модель залишиться універсальною, демонструючи гнучкість підходу.

ДЯКУЮ ЗА УВАГУ