НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Система розпізнавання рукописного тексту з використанням методів комп'ютерного зору»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи асистент	_____ (підпис)	**Денис ВІННІЧУК**
Консультант д.т.н., професор	_____ (підпис)	**Віктор СЕМКО**
Виконав	_____ (підпис)	**Владислав ОЧЕРЕТЯНКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ

(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ

_____Очеретянко Владиславу Володимировичу_____

(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Система розпізнавання рукописного тексту з використанням методів комп'ютерного зору»

Затверджена наказом від «6» січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «25» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту):

1. Кросплатформний програмний застосунок ZenDraw на базі фреймворку Kivy (для забезпечення десктопної та мобільної Android-версій).

2. Вихідний код програмних модулів на мові Python:

- main.py (точка входу та керування життєвим циклом GUI);
- image_processor.py (модуль растрової попередньої обробки штрихів, межового обрізання та нормалізації до матриць 32×32 пікселів);
- recognizer.py (реалізація згорткової нейронної мережі SymbolCNN суто на базі бібліотеки NumPy без зовнішніх ML-фреймворків);
- layout_engine.py (рушій просторового двовимірного компонування дерева символів та керування слотами приєднання);

- history_db.py (підсистема збереження станів та реалізації паттерну Undo/Redo).

3. Конфігураційний файл symbols.txt (опис класів символів, їхніх категорій та доступних слотів). Набір ваг навченої моделі нейромережі. Специфікація формату серіалізації сесій користувача на базі тексту JSON. Демонстраційні графічні матеріали інтерфейсу користувача та матеріали апробації (тези доповідей).

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області: Дослідження існуючих підходів до введення математичних формул (WYSIWYG, LaTeX, OCR) та обґрунтування доцільності розробки локальної інтерактивної системи з механізмом видимих слотів. Формулювання детальних функціональних (малювання, Top-5 корекція, навігація) та нефункциональних вимог (кросплатформність, швидкість reflow) до застосунку ZenDraw.

2. Проєктування інформаційного забезпечення: Розробка інформаційної моделі математичного виразу у вигляді динамічного ієрархічного дерева символів (SymbolNode). Обґрунтування геометрії слотів приєднання (RIGHT, SUPER, SUB, NUMER, DENOM, RADICAND, LIMIT_UP, LIMIT_DOWN) та структури текстового конфігуратора правил. Проєктування схеми серіалізації дерева у формат JSON та механізму знімків станів (Snapshot) для історії дій.

3. Проєктування та реалізація прикладного ПЗ: Розробка архітектури кросплатформного GUI-інтерфейсу на Kivy з підтримкою панорамування, масштабування полотна та перемикання тем. Реалізація конвеєру обробки зображення (межовий прямокутник штрихів, ресайз). Програмна реалізація архітектури нейромережі SymbolCNN на NumPy та алгоритму просторового аналізу (Reflow) для автоматичного розрахунку базових ліній та координат символів.

4. Тестування, впровадження та експлуатація: Проведення модульного тестування геометричних функцій LayoutEngine та інтеграційного тестування

наскрізних сценаріїв створення складних виразів (індекси, дроби, корені). Аналіз особливостей розгортання та роботи застосунку на десктопних ОС та мобільній платформі Android. Визначення обмежень системи та напрямків її подальшого розвитку.

Перелік графічних документів (за необхідності).

Діаграма прецедентів, діаграма послідовності, діаграма діяльності, логічна модель даних, діаграма компонентів, скріншоти інтерфейсу та сценаріїв роботи.

Дата видачі завдання «6» січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Денис ВІННІЧУК**
Консультант д.т.н., професор	_____ (підпис)	**Віктор СЕМКО**
Завдання взяв до виконання	_____ (підпис)	**Владислав ОЧЕРЕТЯНКО**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CNN - Convolutional Neural Network, згорткова нейронна мережа.

GUI - Graphical User Interface, графічний інтерфейс користувача.

JSON - JavaScript Object Notation, текстовий формат обміну структурованими даними.

OCR - Optical Character Recognition, оптичне розпізнавання символів.

Kivy - кросплатформний Python-фреймворк для створення графічних застосунків.

NumPy - бібліотека Python для роботи з масивами та числовими обчисленнями.

LayoutEngine - власний модуль компонування математичних символів у структурований вираз.

SymbolNode - вузол дерева математичного виразу, що відповідає одному символу.

Slot - іменована зона приєднання дочірнього символу до базового символу.

RIGHT - слот горизонтального продовження виразу.

SUPER - слот верхнього індексу.

SUB - слот нижнього індексу.

NUMER - слот чисельника дробу.

DENOM - слот знаменника дробу.

RADICAND - слот підкореневого виразу.

LIMIT_UP - слот верхньої межі великого оператора.

LIMIT_DOWN - слот нижньої межі великого оператора.

Reflow - повторне компонування дерева виразу після зміни його структури.

Top-5 - п’ять найімовірніших результатів розпізнавання символу.

ВСТУП

Сучасне навчальне та наукове середовище активно використовує цифрові інструменти для підготовки конспектів, звітів, методичних матеріалів і наукових текстів. Значна частина таких матеріалів містить математичні вирази, формули, індекси, дроби, корені, грецькі літери, оператори інтегрування та суми. Незважаючи на розвиток текстових редакторів, введення математичних формул у багатьох випадках залишається повільним і неприродним процесом. Користувач змушений або запам’ятовувати команди LaTeX[1], або працювати з вкладеними меню редактора формул, або послідовно обирати елементи у спеціальному вікні.

Особливо помітною ця проблема є під час роботи з мобільними пристроями та планшетами. На сенсорному екрані простіше написати символ пальцем або стилусом, ніж вводити його через клавіатуру. Проте більшість звичних редакторів орієнтована передусім на клавіатурний спосіб роботи. Тому актуальним є створення програмного застосунку, який поєднує природний рукописний спосіб введення з автоматичним розпізнаванням символів і структурованим відображенням математичного виразу.

У межах даної бакалаврської кваліфікаційної роботи розроблено програмний застосунок ZenDraw. Його ідея полягає в тому, що користувач малює математичний символ безпосередньо на полотні, після чого програма розпізнає рукописний начерк, замінює його на друкований символ і розміщує у відповідному місці виразу. Якщо символ намальовано над базовим елементом, він може стати верхнім індексом; якщо під ним - нижнім індексом; якщо всередині зони кореня - підкореневим виразом; якщо у зоні дробу - чисельником або знаменником.

Актуальність роботи зумовлена потребою у зручному, легкому та кросплатформному інструменті для цифрового запису математичних виразів. Такий інструмент може бути корисним студентам, викладачам, школярам, репетиторам, авторам навчальних матеріалів і всім користувачам, які часто

працюють з формулами. На відміну від класичного редактора формул, ZenDraw орієнтований на швидке рукописне введення.

Метою роботи є розробка кросплатформного програмного застосунку для інтерактивного рукописного введення математичних символів, їх автоматичного розпізнавання методами комп'ютерного зору та подальшого структурованого відображення у вигляді математичних виразів.

Об'єктом дослідження є процес цифрового введення та редагування математичних виразів у програмних середовищах. Предметом дослідження є методи й програмні засоби розпізнавання рукописних математичних символів, а також алгоритми їхнього структурного розміщення в математичному виразі.

Для досягнення поставленої мети необхідно виконати такі завдання: проаналізувати предметну область рукописного введення математичних виразів; розглянути існуючі рішення та їх обмеження; сформувати функціональні й нефункціональні вимоги до системи; розробити модель даних математичного виразу; реалізувати модуль попередньої обробки рукописного зображення; реалізувати модуль розпізнавання символів; розробити алгоритм структурного розміщення символів; створити графічний інтерфейс користувача; забезпечити збереження сесій, історію дій, масштабування та роботу на різних пристроях; провести тестування готового програмного застосунку.

Методи розробки включають об'єктно-орієнтоване програмування, комп'ютерний зір, попередню обробку растрових зображень, класифікацію рукописних символів за допомогою згорткової нейронної мережі[2,3,4,5], моделювання структури математичного виразу у вигляді дерева[6,7], а також розробку графічного інтерфейсу користувача з підтримкою сенсорних жестів.

Практичне значення отриманого результату полягає у створенні застосунку, який дозволяє швидше вводити математичні вирази природним рукописним способом. Програма працює як на комп'ютері, так і на мобільних пристроях або планшетах, що робить її зручною для навчального використання.

Розроблена архітектура може бути розширена підтримкою експорту у LaTeX, MathML[8] або формат зображення.

Пояснювальна записка складається зі вступу, чотирьох основних розділів, висновків, списку використаних джерел і додатків. У першому розділі проведено системний аналіз предметної області, сформульовано постановку задачі, вимоги та моделі взаємодії користувача із системою. У другому розділі описано інформаційне забезпечення. У третьому розділі наведено архітектуру й програмну реалізацію основних модулів ZenDraw. У четвертому розділі розглянуто тестування, експлуатацію, вимоги до середовища та перспективи розвитку системи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Постановка завдання полягає в необхідності створення програмного застосунку, який забезпечує швидке та інтуїтивне введення математичних виразів шляхом рукописного малювання символів. Користувач повинен мати можливість намалювати символ на полотні, ініціювати розпізнавання, отримати друковане подання символу та продовжити побудову формули без переходу до складних меню.

Основна складність задачі полягає в тому, що математичний вираз не є простим рядком тексту. У ньому існує двовимірна структура: верхні й нижні індекси, дроби, корені, межі інтегралів і сум, вкладені підвирази, дужки, що можуть розтягуватись за висотою. Тому система повинна не лише класифікувати рукописний символ, але й визначати його роль у структурі формули.

Розроблювана система повинна забезпечувати природність введення. Під природністю у даній роботі розуміється можливість взаємодіяти з формулою наближено до того, як користувач пише її у зошиті: символи вводяться жестом руки, а контекст їхнього розміщення визначається геометрично. Для цього використовується механізм видимих зон-плейсхолдерів.

Вхідними даними для програмного застосунку є координати рукописних штрихів на полотні, зображення обрізаного символу, файл конфігурації математичних символів, ваги навченої нейронної мережі та дані збереженої сесії. Вихідними даними є розпізнаний символ, структурований математичний вираз у внутрішньому поданні, візуальне відображення виразу на полотні, JSON-файл сесії та записи історії дій.

Кінцевим результатом роботи має бути застосунок ZenDraw, який підтримує рукописне введення цифр, латинських і грецьких літер, математичних операторів, символів відношень, дробів, коренів та великих операторів. Додатково система повинна підтримувати панорамування, масштабування, зміну теми інтерфейсу, очищення полотна, збереження й завантаження роботи.

Задача належить до класу інтелектуальних інформаційних систем з елементами комп'ютерного зору. Інтелектуальний компонент представлений модулем розпізнавання рукописних символів, який використовує згорткову нейронну мережу. Інформаційний компонент представлений моделлю даних математичного виразу, конфігурацією символів і механізмом збереження стану.

У межах роботи не ставиться завдання повного семантичного аналізу математичних виразів або перевірки їх математичної правильності. Система не обчислює значення формули та не виконує символьні перетворення. Її завдання полягає у введенні, розпізнаванні, структурному розміщенні та редагуванні математичного запису.

Таблиця 1.1 – Основні вхідні та вихідні дані системи

Група даних	Приклади	Призначення
Вхідні дані	Рукописні штрихи, межовий прямокутник, файл symbols.txt, ваги моделі	Передаються у модулі обробки, розпізнавання та компонування
Проміжні дані	Зображення 32x32, top-5 прогнозів, глобальні координати	Використовуються під час розпізнавання і вставлення символу
Вихідні дані	Структурований вираз, JSON-сесія, оновлене полотно	Відображаються користувачу або зберігаються у файл

1.2 Аналіз предметної області рукописного введення математичних виразів

Предметна область цифрового введення математичних виразів поєднує задачі обробки зображень, розпізнавання рукописних символів, побудови графічного інтерфейсу та математичного набору. З одного боку, користувач очікує, що символи будуть розпізнані так само просто, як текст у звичайному OCR[9]. З іншого боку, математичні символи мають значно більше графічних і структурних особливостей.

Введення формул у текстових редакторах традиційно базується на шаблонах. Користувач спочатку обирає тип конструкції: дріб, корінь, індекс, інтеграл, матрицю або дужки. Після цього він вводить вміст у відповідні порожні поля. Такий спосіб є точним, але він часто перериває потік мислення.

У математичних виразах важливу роль відіграє контекст. Наприклад, знак мінус може бути бінарним оператором віднімання, унарним знаком або горизонтальною рискою дробу. У розробленій системі символ «/» відображається як горизонтальна дробова риска, яка має слоти NUMER і DENOM. Для кореня передбачено слот RADICAND, а для великих операторів - слоти LIMIT_UP і LIMIT_DOWN.

Для рукописного введення важливо враховувати неточність жестів. Користувач може намалювати символ трохи вище або нижче очікуваного місця, різного розміру або з різним нахилом. Тому система використовує не одну точку, а межовий прямокутник намальованих штрихів. Саме цей прямокутник порівнюється з прямокутниками доступних слотів.

Класична задача OCR зазвичай працює зі статичним зображенням, де весь текст уже написано. У ZenDraw процес інтерактивний: користувач додає символи поступово, а структура виразу постійно оновлюється. Це означає, що система повинна швидко перераховувати геометрію вузлів, не створюючи відчутних затримок.

Система належить до однокористувацьких локальних інформаційних систем. Вона не потребує серверної частини і може працювати автономно на персональному комп’ютері або мобільному пристрої. Таке рішення відповідає сценарію навчального використання, коли користувач хоче швидко зробити математичний запис без підключення до мережі.

Перевагою локального підходу є контроль над даними та передбачуваність роботи. Недоліком є те, що розпізнавання повинно виконуватися на ресурсах пристрою користувача, тому модель має бути достатньо компактною. Саме тому в роботі використано реалізацію CNN на NumPy[10,11] без залежності від важких платформ машинного навчання.

Математичний запис має багато винятків, тому система правил повинна бути достатньо гнучкою. Наприклад, грецькі літери можуть мати індекси, оператори зазвичай не повинні їх мати, а великі оператори потребують окремих

меж. Конфігураційний файл дозволяє виразити такі відмінності без складної логіки у кожному місці коду.

1.3 Огляд інформаційних джерел та існуючих рішень

Для визначення місця розроблюваного застосунку було розглянуто декілька груп існуючих рішень. До першої групи належать вбудовані редактори формул у текстових процесорах. Їх перевагою є інтеграція з документом і підтримка багатьох математичних конструкцій. Проте взаємодія часто відбувається через меню, клавіатурні скорочення або спеціальне поле «Рівняння».

Друга група рішень - LaTeX-редактори. LaTeX забезпечує високоякісний математичний набір і є стандартом для наукових публікацій. Однак для ефективної роботи потрібно знати синтаксис команд. Для досвідченого користувача це зручно, але для початківця або для швидкого конспектування поріг входу є високим.

Третя група - системи рукописного розпізнавання математичних виразів. Вони можуть розпізнавати формули зі зображення або з рукописного введення. Такі системи часто використовують складні моделі та серверну обробку. Їх перевагою є висока якість розпізнавання повних виразів, але недоліком може бути залежність від мережі.

Четверта група - графічні редактори й програми для нотаток, які дозволяють писати формули стилусом. Вони добре зберігають рукописний вигляд, але часто не перетворюють запис у структуровані символи. У такому випадку формула залишається малюнком, а не редагованим математичним об’єктом.

Важливою відмінністю ZenDraw є механізм слотів. Більшість звичайних редакторів вимагає спочатку вибрати конструкцію, а потім заповнювати її. У розробленому застосунку конструкція виникає з геометрії взаємодії: користувач малює символ у зоні, яку бачить на екрані.

Під час аналізу аналогів було визначено, що жодна з розглянутих груп рішень повністю не відповідає поставленій меті. Текстові редактори мають сильний механізм набору, але слабку природність введення. LaTeX має високу якість результату, але вимагає знання команд. Графічні нотатники забезпечують свободу письма, але не формують структурованого виразу.

Окремо слід зазначити, що у контексті навчального процесу швидкість введення формули часто важливіша за наявність усіх можливих професійних функцій. Студент, який розв’язує задачу або конспектує лекцію, зазвичай хоче швидко зафіксувати хід думки. Якщо для запису кожного степеня або дробу потрібно відкривати меню, темп роботи сповільнюється. Саме тому ZenDraw орієнтований на сценарій швидкої побудови виразу, а не на повну заміну професійних систем комп’ютерної алгебри.

При розробці системи важливо було знайти баланс між автоматикою та керованістю. Повністю автоматичне розпізнавання всього виразу може бути зручним, але в разі помилки користувачу складно зрозуміти, чому система прийняла саме таке рішення. У ZenDraw кожна дія є локальною: користувач бачить, куди буде вставлено символ, і може контролювати результат. Це підвищує передбачуваність інтерфейсу.

Таблиця 1.2 – Порівняння підходів до введення математичних виразів

Підхід / рішення	**Переваги**	**Недоліки**
Редактор формул у текстовому процесорі	Інтеграція з документом, готові шаблони, точне форматування	Повільне введення, необхідність працювати з меню та вкладеними конструкціями
LaTeX	Висока якість математичного набору, стандарт у наукових текстах	Потрібно знати синтаксис команд, високий поріг входу
Графічні нотатники	Природне письмо стилусом, швидкість конспектування	Формула часто залишається зображенням, а не структурованим об’єктом
Системи розпізнавання повних формул	Можуть перетворювати складні записи у цифровий формат	Можлива залежність від сервера, закритість алгоритмів, складність редагування
ZenDraw	Рукописне введення, локальне розпізнавання, видимі слоти,	Потребує подальшого розширення класів і експорту

Підхід / рішення	Переваги	Недоліки
	структуроване дерево виразу	

1.4 Вимоги до програмного застосунку ZenDraw

Функціональні вимоги визначають, які дії має підтримувати програмна система. Для ZenDraw основною функцією є рукописне введення символів на полотні. Користувач повинен мати можливість малювати штрихи мишею, пальцем або стилусом. Після завершення малювання користувач ініціює розпізнавання, а система вставляє отриманий символ у математичний вираз.

Система повинна підтримувати структурне розміщення символів. Для цього необхідно реалізувати горизонтальне продовження виразу, верхні та нижні індекси, чисельник і знаменник дробу, підкореневу область, а також верхні та нижні межі великих операторів.

Додатковою функціональною вимогою є можливість виправлення помилок розпізнавання. Оскільки нейронна мережа може іноді помилитися, застосунок повинен зберігати декілька найімовірніших прогнозів. Користувач може перемкнути символ на наступний варіант з top-5, не перемальовуючи всю формулу.

Система повинна підтримувати операції редагування: undo, redo, clear, save, load. Операція undo повертає попередній стан виразу, redo повторно застосовує скасовану дію, clear очищає полотно, save зберігає поточну сесію у файл JSON, load відновлює сесію з файлу.

Нефункціональні вимоги стосуються зручності, продуктивності, переносимості та надійності. Інтерфейс повинен бути простим, не перевантаженим кнопками та придатним для роботи як на великому екрані комп’ютера, так і на мобільному пристрої.

Кросплатформність є однією з важливих вимог. Застосунок повинен працювати на настільних операційних системах і мати можливість запуску на

Android. Для цього обрано фреймворк Kivy[12], який підтримує сенсорні події, графічний інтерфейс і пакування мобільних застосунків.

Надійність системи забезпечується збереженням станів у історії та можливістю експорту сесії. При аварійному завершенні або помилці користувач повинен мати змогу відновити роботу з останнього збереженого файлу.

З погляду користувацького досвіду важливими є також візуальні підказки. Сірі placeholder-зони вказують, де саме можна домалювати символ. На відміну від прихованих правил, вони роблять систему зрозумілою навіть без інструкції. Користувач бачить порожнє місце для степеня або знаменника і природно малює символ саме там.

Інтерактивна побудова формули потребує якісної синхронізації між модулями. Якщо розпізнавання повернуло символ, але layout неправильно обробив координати, користувач отримає несподіваний результат. Тому у програмі велика увага приділена єдиним функціям перетворення координат і використанню глобальної системи для всіх геометричних рішень.

Важливо, що layout_engine.py не використовує Kivy і не залежить від графічного середовища. Це означає, що логіку компонування можна перевіряти окремими модульними тестами: створити вузол, додати дочірній символ, виконати reflow і перевірити прямокутники. Такий поділ зменшує ризик помилок і полегшує подальший розвиток.

Таблиця 1.3 – Функціональні вимоги до системи

№	Вимога	Очікуваний результат
F1	Малювання символу на полотні	Користувач створює рукописний штрих мишею, пальцем або стилусом
F2	Розпізнавання символу	Система повертає top-5 прогнозів і вставляє найімовірніший символ
F3	Структурне вставлення	Символ приєднується до RIGHT, SUPER, SUB, NUMER, DENOM, RADICAND або LIMIT-слота

№	Вимога	Очікуваний результат
F4	Виправлення розпізнавання	Користувач перемикає символ між top-5 варіантами
F5	Undo/redo	Користувач повертає попередній або наступний стан
F6	Save/load	Сесія зберігається у JSON і завантажується назад
F7	Панорамування і масштабування	Користувач працює з великим полотном
F8	Day/night режим	Інтерфейс адаптується до умов освітлення

Таблиця 1.4 – Нефункціональні вимоги до системи

Група вимог	Зміст
Зручність	Інтерфейс має бути мінімалістичним, зрозумілим і придатним для сенсорного введення
Продуктивність	Розпізнавання та reflow мають виконуватися без помітних затримок для користувача
Переносимість	Застосунок має працювати на ПК і мати можливість запуску на Android
Надійність	Некоректний ввід не повинен призводити до аварійного завершення
Розширюваність	Нові символи й правила мають додаватися через конфігурацію або окремі модулі

1.5 Моделювання предметної області

Моделювання предметної області виконано з метою формального опису взаємодії користувача із системою, послідовності обробки рукописного введення та внутрішньої структури застосунку. Для цього доцільно використати UML-діаграми[13], оскільки вони дозволяють подати систему з різних точок зору.

На діаграмі прецедентів основним актором є користувач. Він може малювати символ, запускати розпізнавання, виправляти результат, створювати степені й індекси, формувати дроби та корені, виконувати панорамування і масштабування, перемикати тему інтерфейсу, зберігати сесію, завантажувати попередню сесію, очищати полотно, а також виконувати undo/redo.

Діаграма послідовності описує один з основних сценаріїв: користувач малює символ на полотні та запускає розпізнавання. Компонент DrawingCanvas формує зображення рукописного фрагмента і передає його до модуля попередньої обробки. Далі Recognizer виконує класифікацію і повертає список прогнозів.

Діаграма діяльності відображає загальний алгоритм роботи. Спочатку користувач створює рукописний штрих. Якщо зображення коректне, виконується обрізання, нормалізація й розпізнавання. Після отримання прогнозу система перевіряє, чи перекриває рукописний прямокутник один із видимих слотів.

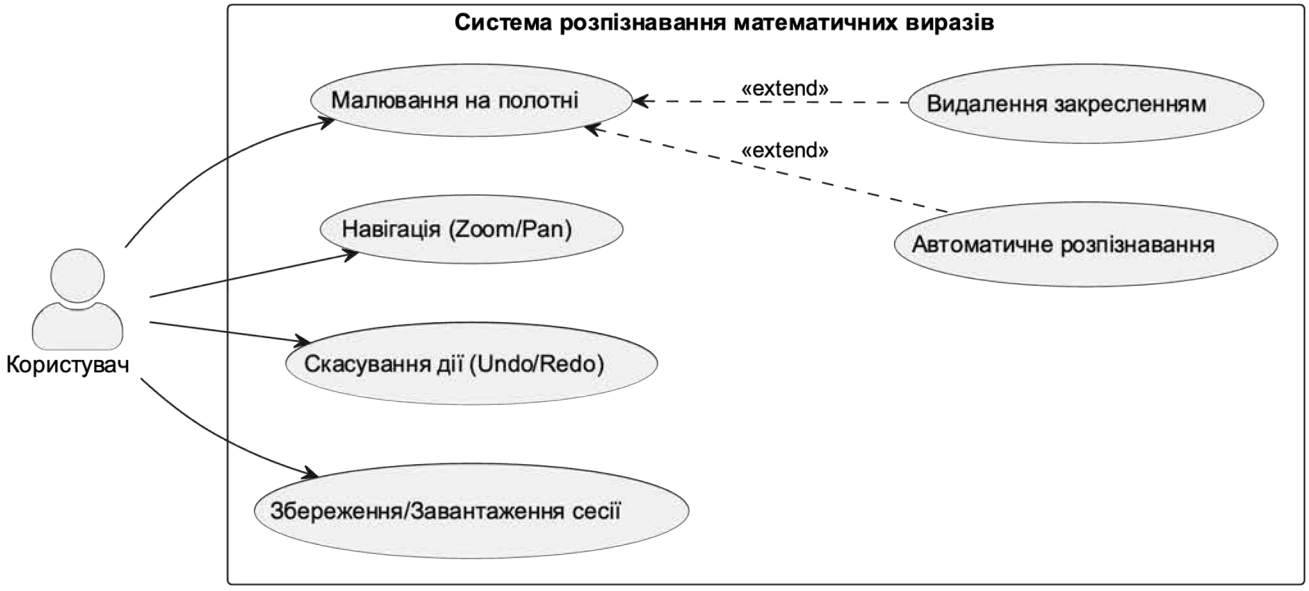

Рисунок 1.1 – Діаграма прецедентів програмного застосунку ZenDraw

На Рис. 1.1 зображено основного актора «Користувач» і межі системи ZenDraw. Центральними прецедентами мають бути «Намалювати символ», «Розпізнати символ», «Побудувати математичний вираз», «Виправити розпізнавання», «Зберегти сесію», «Завантажити сесію», «Скасувати дію» та «Змінити налаштування інтерфейсу».

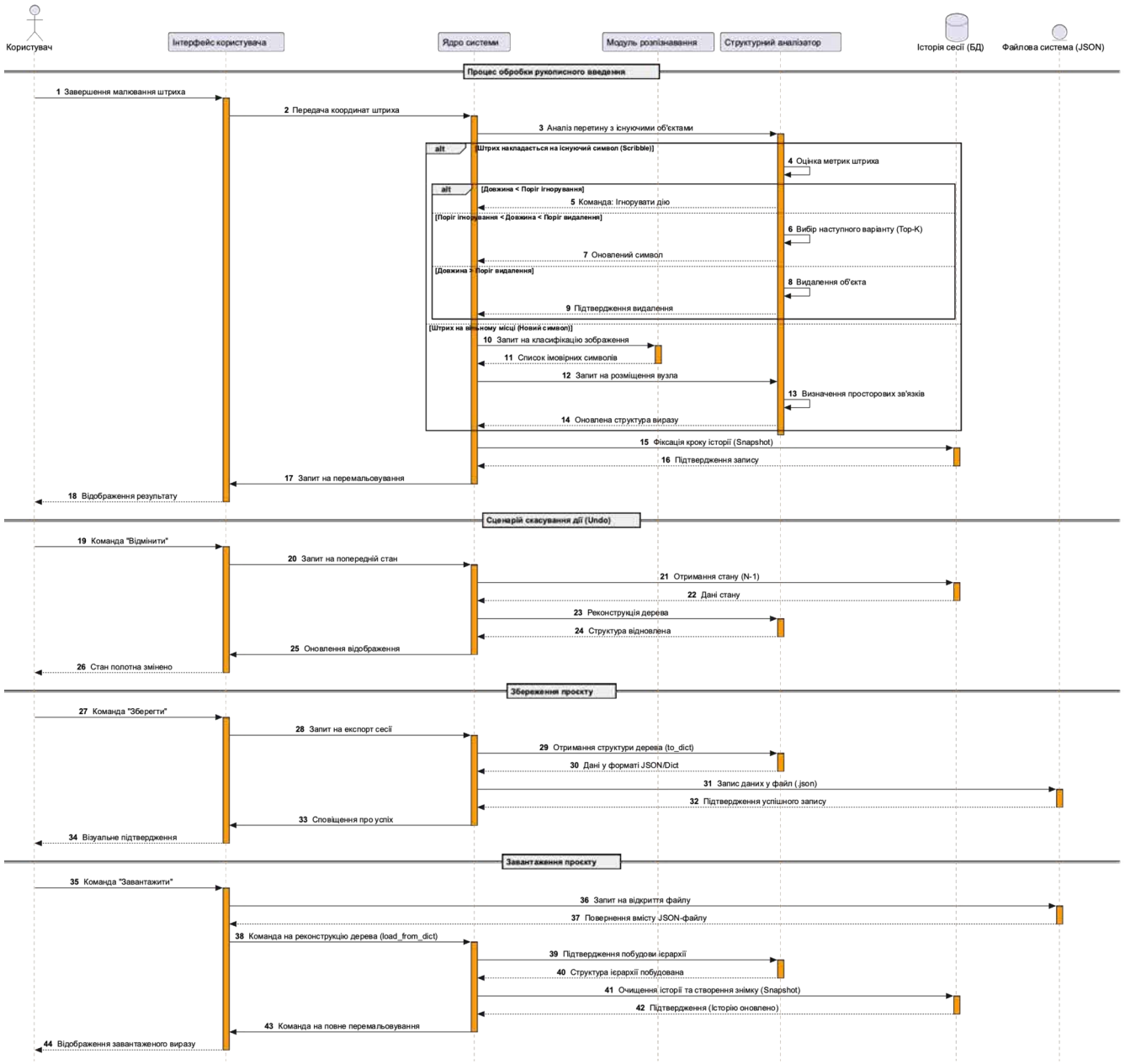

Рисунок 1.2 – Діаграма послідовності розпізнавання та вставлення символу

На Рис. 1.2 зображено хронологію обробки одного символу. Спочатку користувач малює штрих, потім полотно формує обрізане зображення, модуль обробки нормалізує його, розпізнавач повертає top-5, layout-рушій вставляє символ, історія зберігає стан, а інтерфейс перемальовує вираз.

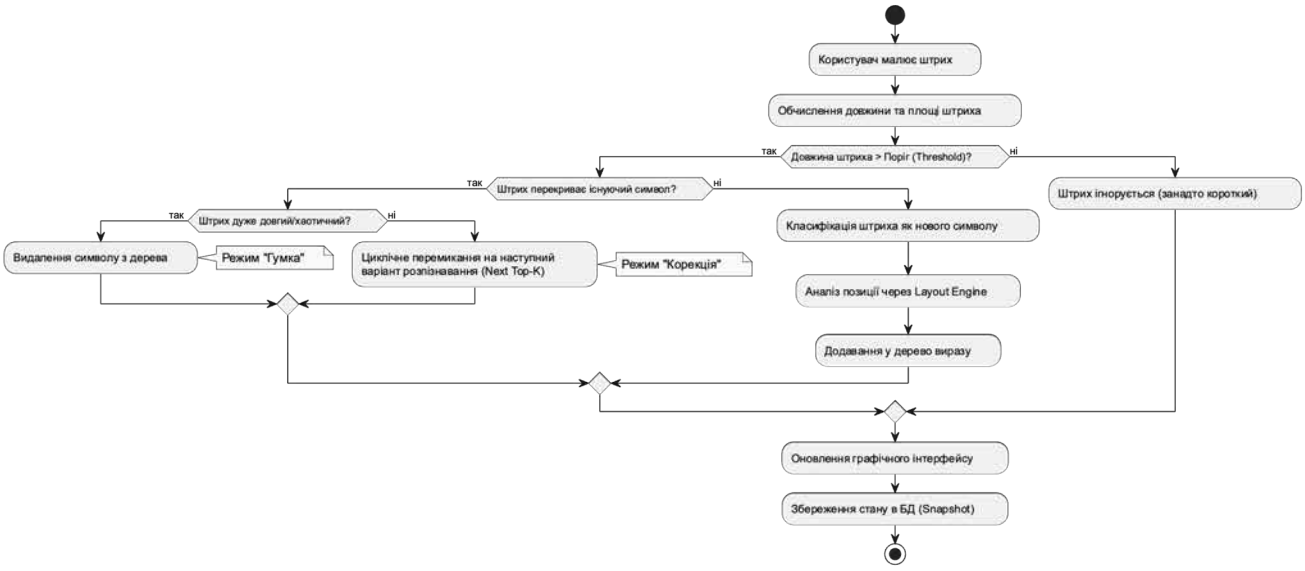

Рисунок 1.3 – Діаграма діяльності основного алгоритму роботи ZenDraw

На Рис. 1.3 показано розгалуження: якщо placeholder-зона перекрита, символ вставляється у відповідний слот; якщо ні, перевіряється RIGHT-ланцюг; якщо і він не підходить, створюється новий кореневий вузол.

2 ПРОЄКТУВАННЯ І РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Логічна модель даних

Інформаційне забезпечення системи визначає, які дані використовуються програмним застосунком, як вони організовані, у яких форматах зберігаються та яким чином передаються між модулями. Для ZenDraw ключовими інформаційними об’єктами є рукописний штрих, растрове зображення символу, прогноз розпізнавання, вузол математичного виразу, слот приєднання, дерево виразу, історія станів через базу даних SQlite[14] і файл сесії.

Логічна модель даних системи базується на поданні математичного виразу у вигляді дерева. Кожен вузол дерева відповідає одному математичному символу. Вузол має текстове значення, прямокутник розміщення, масштаб, категорію, список доступних слотів, посилання на батьківський вузол і набір дочірніх вузлів.

Такий спосіб подання є зручним, оскільки він відображає природну ієрархію математичного запису. Наприклад, у виразі x^2 число 2 є дочірнім вузлом символу x у слоті SUPER. У дробі чисельник і знаменник є дочірніми вузлами дробової риски. У корені підкореневий вираз є дочірнім вузлом символу √ у слоті RADICAND.

Кожен вузол має прямокутник rect, який задає позицію та розмір символу у глобальних координатах. Глобальна система координат використовується для компонування формул і не залежить від поточного масштабу екрана. Окремо існує екранна система координат Kivy, у якій відображаються жести користувача.

Модель даних передбачає збереження не лише фінального символу, а й списку прогнозів розпізнавання. Поля predictions і pred_index дають можливість перемикатися між альтернативними результатами. Це важливо для практичної роботи, оскільки рукописні символи часто неоднозначні.

Логічна модель також включає конфігураційний файл symbols.txt. У ньому задається відповідність між ідентифікатором класу моделі та графічним символом, а також перелік слотів, доступних для цього символу. Завдяки цьому правила компонування винесено з коду в окремий файл.

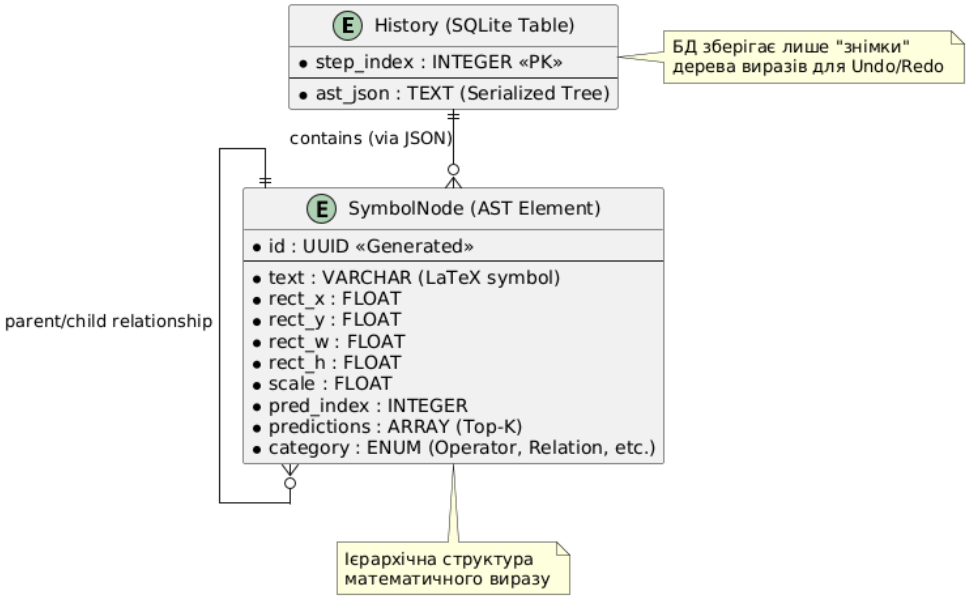

Рисунок 2.1 – Логічна модель даних програмного застосунку

У логічній моделі даних потрібно відобразити, що сесія містить один або декілька кореневих вузлів, кожен вузол має дочірні вузли у слотах, а також пов’язаний зі списком прогнозів розпізнавання. Окремим елементом варто показати конфігурацію символів, з якої завантажуються правила доступних слотів.

2.2 Модель математичного виразу та дерево символів

Модель математичного виразу в ZenDraw побудована навколо класу SymbolNode. Цей клас представляє окремий символ і всю інформацію, необхідну для його відображення та взаємодії з іншими символами. Основними атрибутами є text, rect, scale, parent, parent_slot, slots, predictions, pred_index, category і baseline_offset.

Категорія символу визначає його поведінку в layout-рушії. У системі виділяються базові символи, великі оператори, корені, дужки, оператори та відношення. Базові символи, наприклад літери й цифри, зазвичай мають верхній і нижній індекс. Великі оператори можуть мати межі зверху та знизу.

Слоти є центральним елементом інформаційної моделі. Слот RIGHT відповідає горизонтальному продовженню запису. Слот SUPER використовується для верхнього індексу, SUB - для нижнього індексу. Слоти NUMER і DENOM описують чисельник і знаменник дробу.

Кожен слот має геометричну зону очікування - placeholder. Ця зона відображається на екрані як світлий прямокутник. Якщо користувач малює новий символ усередині цієї області, система сприймає це як намір вставити символ саме в цей слот.

Застосування слотів дозволяє уникнути надмірної складності повного розпізнавання математичної структури. Система не повинна самостійно вгадувати, що користувач мав на увазі у всьому виразі. Вона пропонує видимі зони, а користувач підтверджує намір своїм жестом.

Дерево виразу може містити кілька кореневих вузлів. Це необхідно, оскільки користувач може створити декілька незалежних формул або фрагментів на одному полотні. Кожен кореневий вузол зберігається в списку roots, а всі вузли незалежно від ієрархії зберігаються в списку symbols.

Під час рендерингу математичного виразу необхідно враховувати не лише координати прямокутника, але й базову лінію символів. Символи різної висоти мають вирівнюватися так, щоб вираз виглядав природно. Для цього у layout-рушії використовуються спрощені метрики шрифту: ширина символу, висота, x-height, ascender і descender.

Для мобільної роботи особливо важливим є уникнення блокувань інтерфейсу. Якщо розпізнавання або reflow виконуються занадто довго, користувач втрачає відчуття прямої взаємодії. Тому модель має невеликий

вхідний розмір 32x32, а layout-алгоритм працює з простими прямокутниками і деревами, що забезпечує прийнятну швидкодію.

Таблиця 2.1 – Основні поля вузла SymbolNode

Поле	Тип / приклад	Призначення
text	рядок, наприклад x або √	Графічний символ, який відображається на полотні
rect	Rect(x, y, w, h)	Позиція і розмір символу у глобальних координатах
scale	дійсне число	Масштаб символу відносно базового розміру
parent	посилання на SymbolNode	Батьківський вузол у дереві
parent_slot	SUPER, SUB, RIGHT тощо	Назва слота, у якому знаходиться вузол
slots	словник	Дочірні вузли за назвами слотів
predictions	список символів	Top-5 результатів розпізнавання
pred_index	ціле число	Поточний вибраний прогноз
category	BASE, ROOT, LARGE_OP	Категорія символу для визначення поведінки

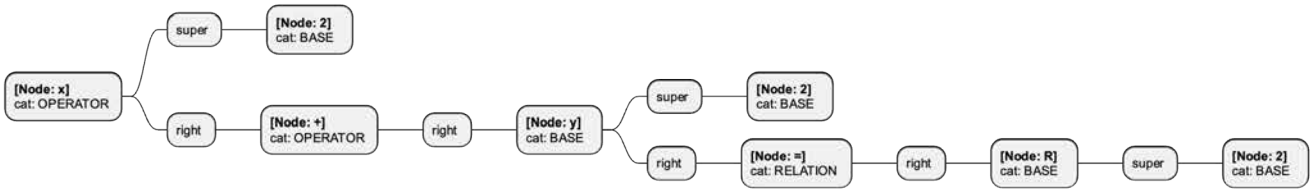

Рисунок 2.2 – Приклад дерева математичного виразу

2.3 Конфігураційне забезпечення математичних символів

Для того щоб система могла не лише розпізнати символ, а й зрозуміти можливі способи його приєднання до математичного виразу, у програмі використовується конфігураційний файл symbols.txt. Він виконує роль довідника математичних символів і правил їх структурної поведінки. На відміну від простого списку класів нейронної мережі, цей файл містить зв’язок між числовим ідентифікатором класу, графічним знаком та набором допустимих слотів.

Формат одного рядка має вигляд: id: glyph | slot1 slot2 ..., де id відповідає класу розпізнавання, glyph є Unicode-представленням[15] символу, а після вертикальної риски вказуються слоти, доступні для цього символу. Якщо слотів не вказано, символ використовується лише як елемент горизонтального запису або застосовує стандартні правила своєї категорії.

Такий підхід дозволяє змінювати поведінку системи без переписування основного програмного коду. Наприклад, для літер, цифр і грецьких змінних дозволені верхні та нижні індекси, для кореня доступний слот radicand, для дробової риски задаються чисельник і знаменник, а для великих операторів, таких як інтеграл або сума, передбачаються межі зверху і знизу.

```
113: x | super sub
114: y | super sub
72: 2 | super sub
960: √ | radicand
922: / | numer denom
183: ∫ | super sub limit_up limit_down
195: - |
```

Рисунок 2.3 – Приклад формату конфігураційного файлу

Таблиця 2.2 – Призначення основних слотів математичного виразу

Назва слота	Призначення	Приклад використання
RIGHT	Горизонтальне продовження виразу праворуч від поточного символу	x + y = 0
SUPER	Верхній індекс або степінь	x^2, λ^3
SUB	Нижній індекс	a_1, x_i
NUMER	Чисельник дробу	2/x
DENOM	Знаменник дробу	2/x
RADICAND	Підкореневий вираз	√x
LIMIT_UP	Верхня межа великого оператора	∑ з верхньою межею
LIMIT_DOWN	Нижня межа великого оператора	∫ з нижньою межею

Інформація зі symbols.txt використовується під час ініціалізації LayoutEngine. Функція завантаження конфігурації розбирає рядки файлу, відкидає коментарі, формує словник відповідності класів до символів та словник відповідності символів до набору слотів. Завдяки цьому розпізнавач і рушій компонування мають спільну інформаційну основу.

Важливо, що RIGHT-слот є неявним і доступний для кожного символу. Це потрібно для звичайного лінійного запису виразу. Додаткові слоти визначаються окремо, щоб оператори на кшталт + або = не отримували степені випадково, а літери та цифри могли коректно використовуватися як основи для індексів.

Використання конфігураційного файла також спрощує подальше розширення системи. Для додавання нового символу достатньо навчити або підключити відповідний клас розпізнавання та додати рядок у symbols.txt із потрібними слотами. Це робить інформаційне забезпечення гнучким і незалежним від конкретного набору математичних знаків.

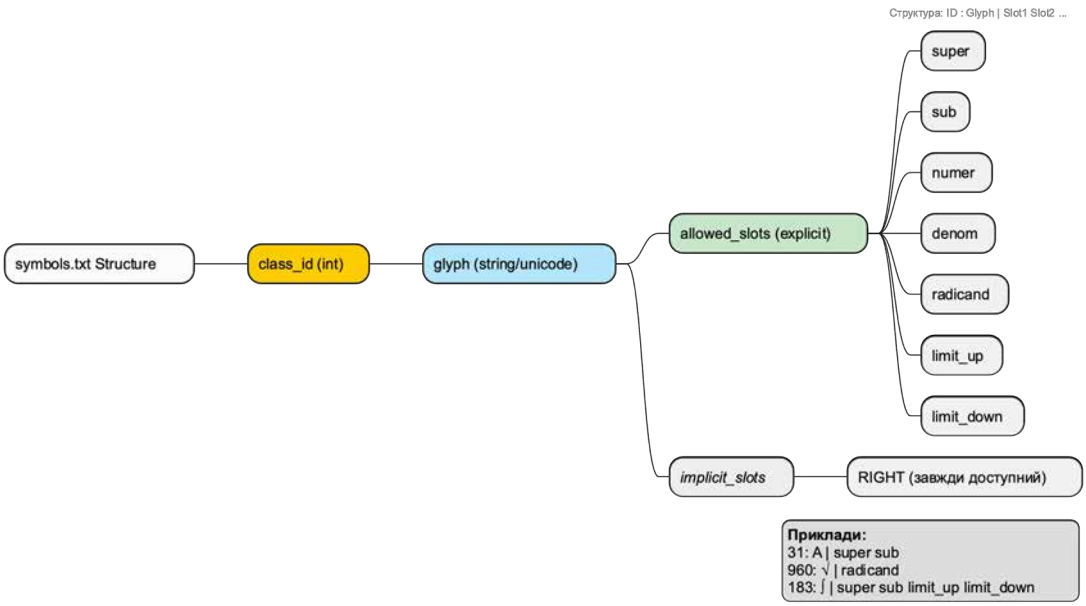

Рисунок 2.4 – Структура конфігураційного забезпечення математичних символів

2.4 Формат збереження стану та історія дій

Для практичного використання застосунку недостатньо лише побудувати математичний вираз на екрані. Користувач повинен мати можливість скасувати помилкову дію, повторити її, зберегти поточну роботу та відновити її пізніше. Тому у системі передбачено механізм серіалізації стану виразу та історії дій.

Стан виразу може бути поданий у вигляді JSON-структури. У ній зберігаються кореневі вузли, текст символів, координати прямокутників, масштаб, передбачені нейронною мережею альтернативи та зв’язки між батьківськими й дочірніми елементами. Такий формат є зручним для людського читання, компактним і зручним для локального збереження.

Механізм undo/redo працює за принципом збереження послідовності станів. Після кожної значущої операції, наприклад додавання символу, зміни розпізнаного варіанта або очищення полотна, поточний стан серіалізується. Під час скасування дії система повертається до попереднього стану, а під час повторення — до наступного.

```
{
  "roots": [
    {
      "text": "x",
```

```
      "rect": [120, 180, 31, 38],
      "scale": 1.0,
      "slots": {
        "super": {
          "text": "2",
          "rect": [152, 142, 19, 26],
          "scale": 0.7,
          "slots": {}
        },
        "right": {
          "text": "+",
          "rect": [190, 182, 28, 34],
          "scale": 1.0,
          "slots": {}
        }
      }
    }
  ]
}
```

Рисунок 2.5 – Приклад формату дерева математичного виразу в файл JSON

Використання JSON також має перевагу під час тестування та відлагодження. Розробник може переглянути файл сесії у звичайному текстовому редакторі й перевірити, чи правильно сформовано дерево виразу. Це особливо важливо для математичних структур, у яких помилка може бути не в символі, а у неправильному батьківському вузлі або неправильному слоті.

У дипломній роботі цей механізм розглядається як частина інформаційного забезпечення, оскільки він описує спосіб представлення, збереження та відновлення інформації про математичний вираз. Програмна реалізація вже працює на рівні локального застосунку, але такий самий формат можна використати для майбутнього експорту, синхронізації або перетворення у LaTeX.

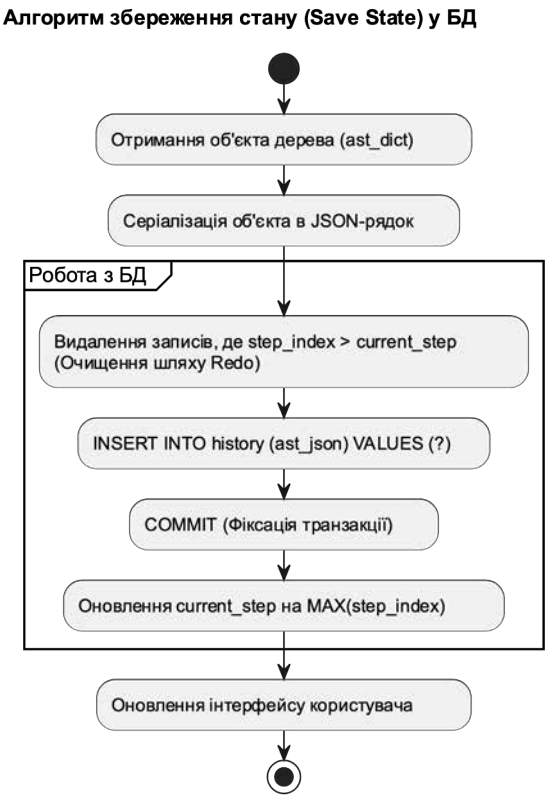

Рисунок 2.6 – Модель збереження стану математичного виразу

2.5 Інформаційні потоки системи

Інформаційні потоки ZenDraw можна поділити на три групи: потоки користувацького введення, потоки обробки зображення та потоки структурного відображення. Користувацьке введення формується у вигляді координат штрихів, отриманих від миші, стилуса або сенсорного екрана. Далі ці координати перетворюються у растрове зображення для розпізнавання.

Потік обробки зображення включає обрізання за межовим прямокутником, нормалізацію розміру, перетворення у відтінки сірого, бінаризацію та підготовку масиву 32×32 пікселі. Після цього масив передається у нейронну мережу SymbolCNN, яка повертає список найбільш імовірних символів.

Потік структурного відображення починається після вибору символу. Новий вузол передається в LayoutEngine разом із прямокутником рукописного вводу. Рушій перевіряє перетин цього прямокутника з порожніми слотами вже

наявного виразу. Якщо перетин достатній, символ приєднується до відповідного слота; якщо ні, створюється новий незалежний корінь.

Після кожної зміни виконується перерахунок геометрії: оновлюються прямокутники символів, координати дочірніх вузлів, placeholder-зони та загальні межі піддерев. Результат передається модулю відображення, який очищує попередній рендер і малює оновлений математичний вираз на полотні.

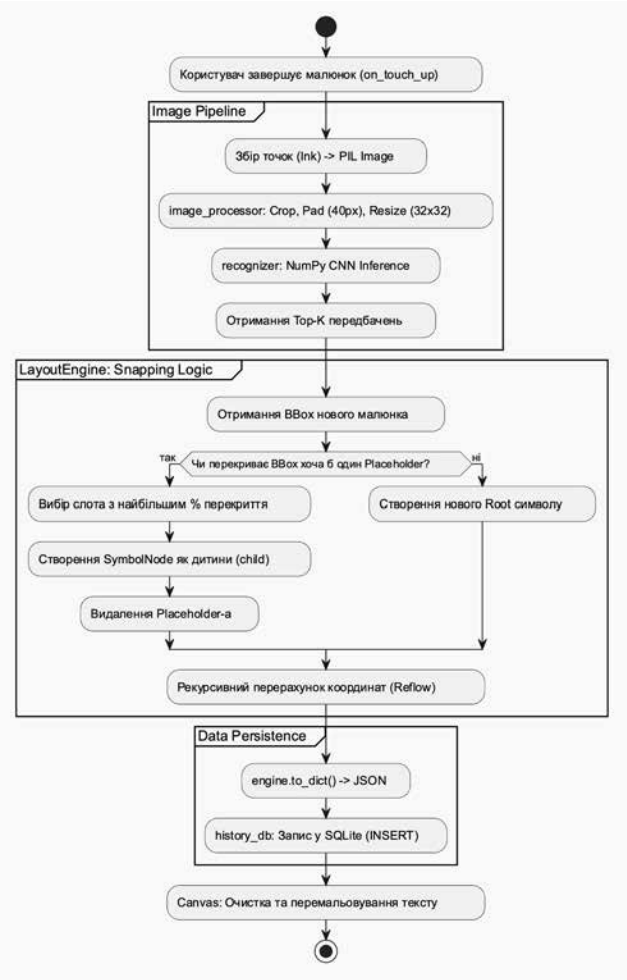

Рисунок 2.7 – Інформаційні потоки програмного застосунку ZenDraw

3 ПРОЄКТУВАННЯ І РОЗРОБКА ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна архітектура програмного застосунку

Програмний застосунок ZenDraw побудований за модульною архітектурою. Кожен основний етап роботи системи винесено в окремий компонент: графічний інтерфейс, обробка рукописного введення, попередня обробка зображення, розпізнавання символу, структурне компонування виразу та збереження історії дій. Такий поділ спрощує супровід програми та дозволяє змінювати окремі модулі незалежно один від одного.

Головний модуль main.py відповідає за ініціалізацію застосунку, створення Kivy-віджетів, обробку подій миші та сенсорного екрана, відображення символів на полотні, взаємодію з LayoutEngine, збереження і завантаження сесій. У цьому модулі також реалізовано логіку меню налаштувань, перемикання теми, плаваючої кнопки розпізнавання та жестів.

Модуль image_processor.py виконує функції, подібні до мінімального набору Pillow[16], але реалізований на чистому Python[17]. Завдяки цьому застосунок не залежить від важких C-розширень, що є важливою умовою для збирання Android-версії. Модуль містить об’єкт Image, засоби малювання, зміну розміру, конвертацію кольорів та прості фільтри.

Модуль recognizer.py реалізує згорткову нейронну мережу SymbolCNN на базі NumPy. На відміну від типового підходу з використанням PyTorch або TensorFlow під час виконання, у ZenDraw нейронна мережа виконується без зовнішнього ML-рушія. Це зменшує залежності, спрощує інсталяцію та підвищує переносимість.

Модуль layout_engine.py є центральним для структурного компонування формули. Він описує геометричні примітиви, вузли символів, слоти, правила приєднання, перерахунок позицій, побудову дробів, коренів, степенів та індексів. Саме цей компонент перетворює набір розпізнаних символів на структурований математичний вираз.

Таблиця 3.1 – Основні програмні модулі ZenDraw

Модуль	Призначення	Ключові функції
main.py	Головний модуль застосунку та графічного інтерфейсу	події, жести, меню, рендер, save/load
layout_engine.py	Рушій математичного компонування	слоти, вузли, reflow, bounds, placeholder-зони
recognizer.py	Розпізнавання символів	CNN на NumPy, softmax, top-5 predictions
image_processor.py	Попередня обробка зображення	crop, resize, convert, filter, draw
symbols.txt	Конфігурація символів	id, glyph, allowed slots
history_db.py	Історія дій і сесії	undo/redo, серіалізація станів

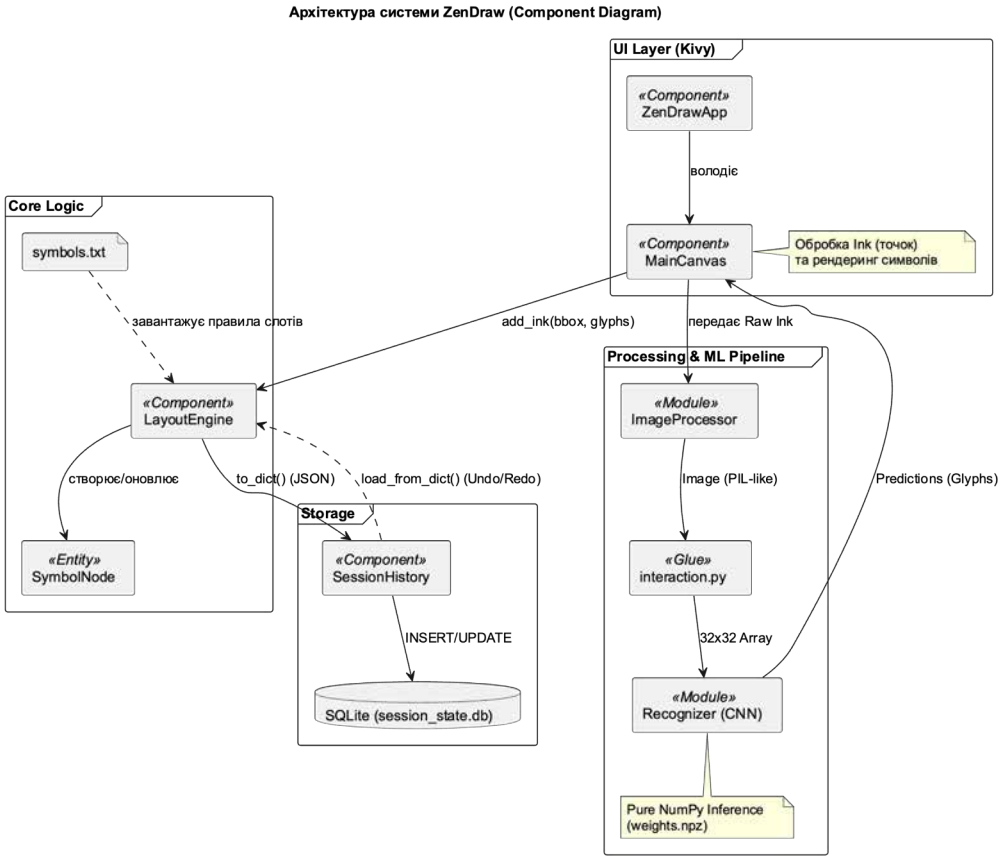

Рисунок 3.1 – Компонентна архітектура програмного застосунку ZenDraw

Архітектура побудована так, щоб графічний інтерфейс не містив усієї математичної логіки. DrawingCanvas відповідає за збирання штрихів і відображення результату, але рішення про те, де саме має бути розташований символ, приймає LayoutEngine. Розпізнавач також не знає про структуру формули, а лише повертає список кандидатів.

Такий розподіл відповідальності відповідає принципам об’єктно-орієнтованого проєктування: кожен компонент має чітку зону відповідальності, а взаємодія між ними відбувається через прості дані: зображення, текст символу, координати прямокутника та дерево вузлів.

3.2 Реалізація графічного інтерфейсу користувача

Графічний інтерфейс ZenDraw реалізовано за допомогою фреймворку Kivy, який підтримує розробку кросплатформних застосунків. Це дало можливість орієнтувати програму не лише на настільний комп’ютер, а й на сенсорні пристрої: телефони та планшети. Основна взаємодія користувача відбувається через велике полотно для рукописного введення.

У ZenDraw використано мінімалістичний підхід до інтерфейсу. Користувач бачить переважно полотно, а допоміжні елементи з’являються лише тоді, коли вони потрібні. Це відповідає початковій ідеї розробки: зробити запис математичних виразів більш природним, ніж введення через клавіатуру або складне вікно формул у текстовому редакторі.

На полотні користувач малює символ мишею, стилусом або пальцем. Координати штрихів зберігаються в екранній системі координат Kivy. Після завершення малювання межі штрихів використовуються для створення растрового зображення, яке передається у модуль розпізнавання.

Інтерфейс підтримує денну та нічну теми. У денному режимі полотно світле, а символи промальовуються чорним кольором. У нічному режимі фон темний, а символи білі. Це важливо для комфортної роботи користувача в різних умовах освітлення.

Для запуску розпізнавання передбачено два основні способи: натискання пробілу на комп’ютері або натискання плаваючої круглої кнопки на сенсорному пристрої.

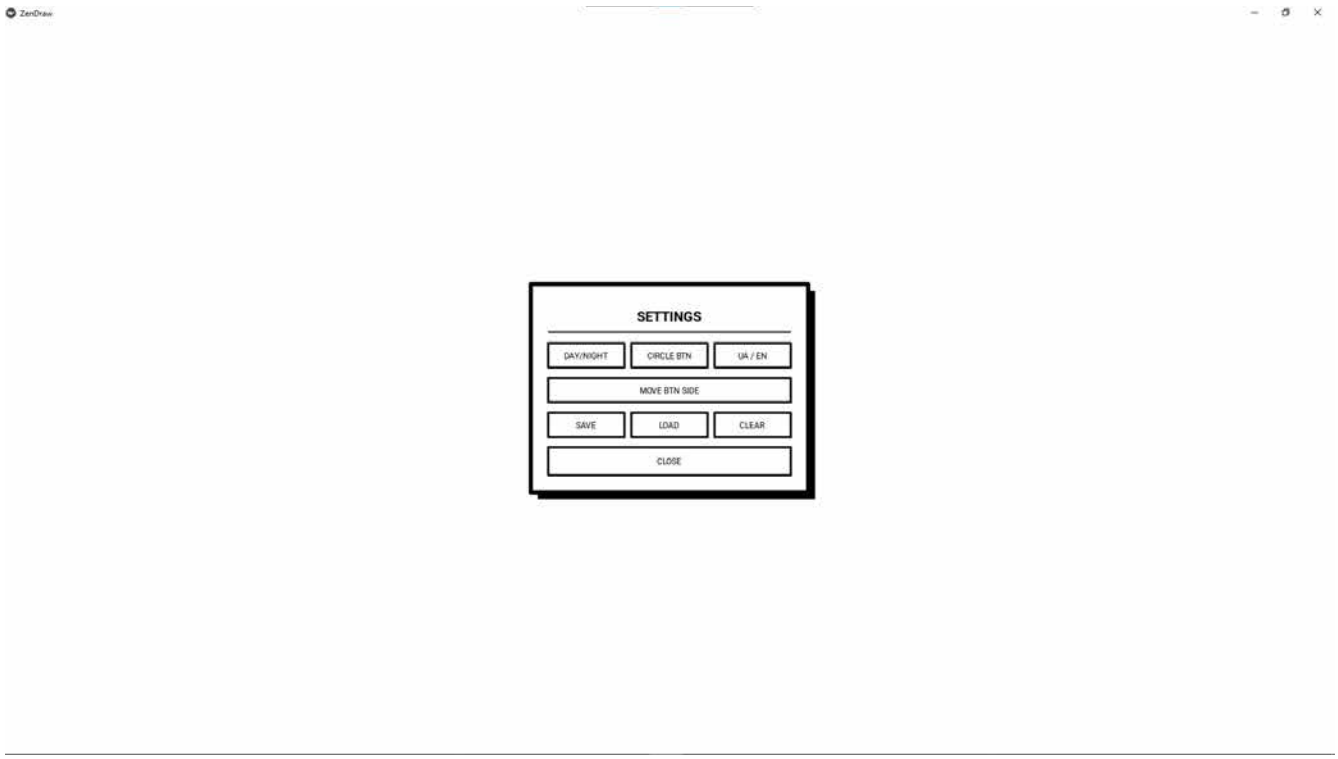

Рисунок 3.2 – Панель налаштувань ZenDraw

3.3 Попередня обробка рукописного зображення

Попередня обробка зображення є необхідним етапом перед передаванням символу до нейронної мережі. Рукописний штрих, який користувач виконує на полотні, може мати різний розмір, положення, товщину та співвідношення сторін. Для стабільного розпізнавання всі такі зображення потрібно привести до єдиного формату.

Першим кроком є визначення межового прямокутника рукописного вводу. Система аналізує мінімальні та максимальні координати точок штриха і формує область, яка містить лише намальований символ. Це дозволяє відкинути зайві порожні ділянки полотна та зосередити розпізнавання на самому символі.

Другим кроком є перетворення зображення у формат відтінків сірого. Для нейронної мережі достатньо одного каналу яскравості, тому RGB-подання не є необхідним. Це зменшує обсяг даних і спрощує обчислення.

Третім кроком є нормалізація розміру. Архітектура SymbolCNN очікує зображення розміром 32×32 пікселі. Тому обрізаний символ масштабується із

збереженням загальної форми. Якщо потрібно, навколо символу додаються поля, щоб уникнути деформації.

Четвертим кроком є нормалізація значень пікселів. Зображення переводиться у числовий масив, значення якого масштабуються до діапазону, придатного для подавання на вхід нейронної мережі. У тезах до роботи описано нормалізацію за середнім значенням 0.5 та стандартним відхиленням 0.5.

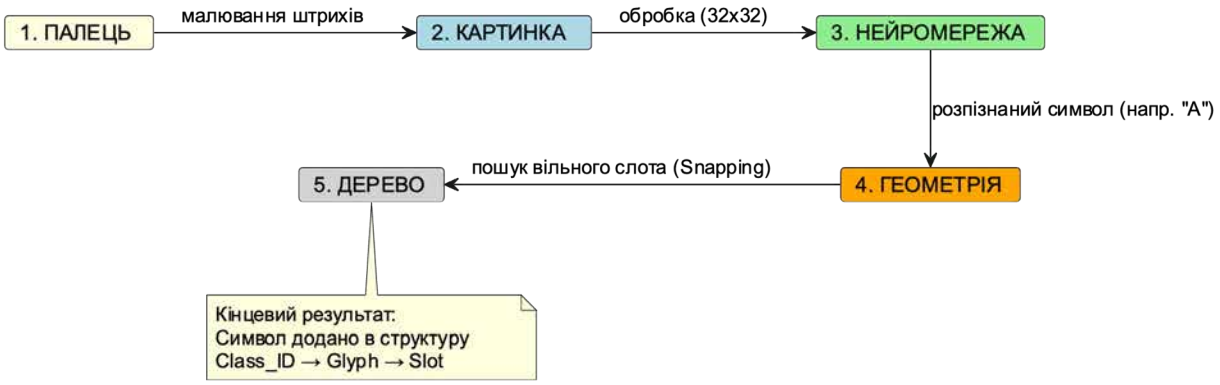

Рисунок 3.3 – Етапи попередньої обробки рукописного символу

```
def prepare_symbol_image(strokes):
    bbox = calculate_bounding_box(strokes)
    cropped = canvas_image.crop(bbox)
    gray = cropped.convert("L")
    square = pad_to_square(gray)
    resized = square.resize((32, 32))
    array = normalize(resized)
    return array
```

Рисунок 3.4 – Код функції для підготовки зображення перед тим як надати CNN моделі

Окремою особливістю проєкту є те, що необхідні операції роботи із зображеннями реалізовано у власному модулі image_processor.py. Це знижує залежність від сторонніх пакетів, які можуть мати проблеми під час збирання мобільної версії. У межах дипломної роботи цей модуль можна розглядати як допоміжний рівень сумісності для операцій crop, resize, convert, paste, point та filter.

Незважаючи на простоту окремих операцій, саме попередня обробка значною мірою впливає на якість розпізнавання. Якщо символ буде обрізано неправильно або надто сильно масштабовано, нейронна мережа може сплутати його з іншим класом. Тому у програмі важливо зберігати відступи навколо символу і не втрачати характерні ознаки рукописної форми.

3.4 Реалізація модуля розпізнавання символів

Модуль recognizer.py реалізує згорткову нейронну мережу SymbolCNN. Його основне завдання полягає в тому, щоб отримати підготовлене зображення 32×32 пікселі та визначити, до якого класу математичних символів воно належить. Результатом роботи є не лише один символ, а список найбільш імовірних варіантів, що дозволяє реалізувати механізм Top-5.

Архітектура мережі складається з кількох згорткових блоків. Кожен блок виділяє локальні ознаки зображення: лінії, кути, перетини, заокруглення та інші характерні елементи рукописного символу. Перші шари працюють з простішими ознаками, а глибші шари формують більш абстрактне представлення символу.

У реалізації використовується NumPy. Це означає, що прямий прохід нейронної мережі виконується за допомогою операцій над масивами без використання PyTorch або ONNX Runtime під час виконання програми. Такий підхід ускладнює реалізацію низькорівневих операцій, але спрощує розгортання застосунку на різних платформах.

Низькорівневі функції модуля виконують згортку, max pooling, ReLU, повнозв'язні шари та softmax. Ваги моделі зберігаються у форматі .npz і завантажуються під час ініціалізації. На етапі інференсу dropout не використовується, оскільки він потрібний лише під час навчання.

Таблиця 3.2 – Узагальнена архітектура SymbolCNN

Блок	Операції	Розмір/призначення
Conv block 1	Conv2d + BatchNorm + ReLU + MaxPool	виділення базових локальних ознак
Conv block 2	Conv2d + BatchNorm + ReLU + MaxPool	ускладнення ознак і зменшення розміру карти
Conv block 3	Conv2d + BatchNorm + ReLU + MaxPool	формування стійких ознак символу
Conv block 4	Conv2d + BatchNorm + ReLU	високорівневе представлення
Classifier	Flatten + Linear + ReLU + Linear	класифікація символу
Softmax	Нормалізація логітів	ймовірності класів і Top-5

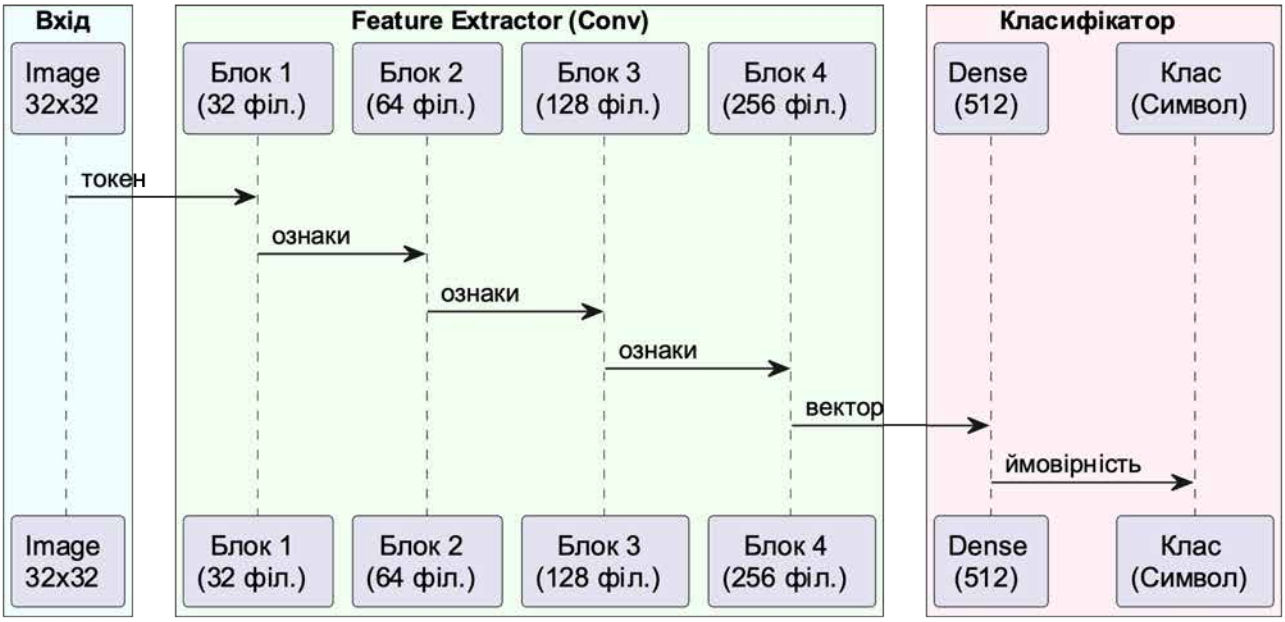

Рисунок 3.5 – Архітектура згорткової нейронної мережі SymbolCNN

```
def recognize(image_32x32):
```

```
    x = image_32x32.reshape(1, 1, 32, 32)
    logits = forward_symbol_cnn(x, weights)
    probabilities = softmax(logits)
    top_indices = argsort(probabilities)[-5:]
    return [class_to_symbol[i] for i in top_indices]
```

Рисунок 3.6 – Код функції для розпізнавання символу на зображенні за допомогою CNN моделі

Повернення кількох найкращих варіантів є важливою практичною функцією. У рукописному введенні деякі символи можуть бути схожими, наприклад 2 і z, 0 і O, x і ×, λ і інші грецькі символи. Якщо система помилилася, користувач може перемкнутися на інший варіант без повторного введення всього виразу.

Таким чином, модуль розпізнавання не замінює повністю користувача, а працює як інтелектуальний помічник. Він пропонує найбільш імовірні варіанти, а кінцева взаємодія залишається контрольованою і зрозумілою.

3.5 Розробка LayoutEngine для структурного компонування виразів

LayoutEngine є ключовою частиною ZenDraw, оскільки саме він відповідає за перетворення розпізнаних символів у математично структурований вираз. Без цього модуля програма могла б лише замінювати рукописні знаки на друковані, але не могла б формувати степені, індекси, дроби, корені та межі операторів.

Основною одиницею структури є SymbolNode. Кожен вузол містить текст символу, прямокутник його розташування, масштаб, список прогнозів, індекс поточного прогнозу, категорію символу та словник дочірніх слотів. Завдяки цьому один символ може мати вкладені елементи, а весь вираз подається як дерево або ліс дерев.

Для розміщення нового символу рушій використовує модель порожніх слотів. Кожний незаповнений слот проєктується у вигляді прямокутної placeholder-зони. Коли користувач малює новий символ у межах такої зони, система обчислює частку перекриття між рукописним прямокутником і

прямокутником слота. Якщо перекриття перевищує заданий поріг, символ приєднується саме до цього слота.

У коді використовується поріг SNAP_OVERLAP_THRESHOLD. Його сенс полягає в тому, що користувач повинен справді намалювати символ у зоні підказки, а не просто поруч із нею. Це робить поведінку системи передбачуваною: якщо користувач хоче створити степінь, він малює символ у верхньому placeholder; якщо хоче продовжити вираз, малює праворуч.

Після приєднання символу LayoutEngine виконує reflow. Під час цього процесу перераховується положення основного символу, дочірніх елементів та наступних символів у ланцюгу RIGHT. Для дробу формується горизонтальна риска, для кореня будується підкореневий простір, для верхніх та нижніх індексів застосовується зменшений масштаб.

Рисунок 3.7 – Алгоритм приєднання символу до математичної структури

```
def add_symbol(glyph, ink_rect):
    node = SymbolNode(glyph, ink_rect)
    candidate_slot = find_best_overlapped_placeholder(ink_rect)


    if candidate_slot is not None:
        parent, slot_name = candidate_slot
        parent.set_child(slot_name, node)
    else:
        roots.append(node)


    reflow()
    return node
```

Рисунок 3.8 – Код функції додавання символа до дерева математичного виразу

Таблиця 3.3 – Категорії символів у LayoutEngine

Категорія	**Приклади**	**Особливості компонування**
BASE	x, y, R, λ, 2	може мати верхній і нижній індекси
OPERATOR	+, -, ×, ÷	використовується переважно у горизонтальному записі
RELATION	**=**, **<**, **>**, **≈**, **≠**	розділяє частини виразу
ROOT	√	має підкореневий слот radicand
LARGE_OP	∫, ∑, ∏	може мати межі зверху та знизу
FENCE	(,), [,], |	може масштабуватися відповідно до вкладеного виразу

Механізм слотів дозволяє реалізувати природну взаємодію з формулою. Користувачеві не потрібно натискати окрему кнопку для створення степеня або дробу. Достатньо намалювати символ у відповідній зоні, а програма визначить контекст сама.

На поточному етапі найважливішими структурами є горизонтальний ланцюг, степінь, індекс, дріб, корінь та межі великих операторів. Саме ці конструкції покривають значну частину типових навчальних математичних записів.

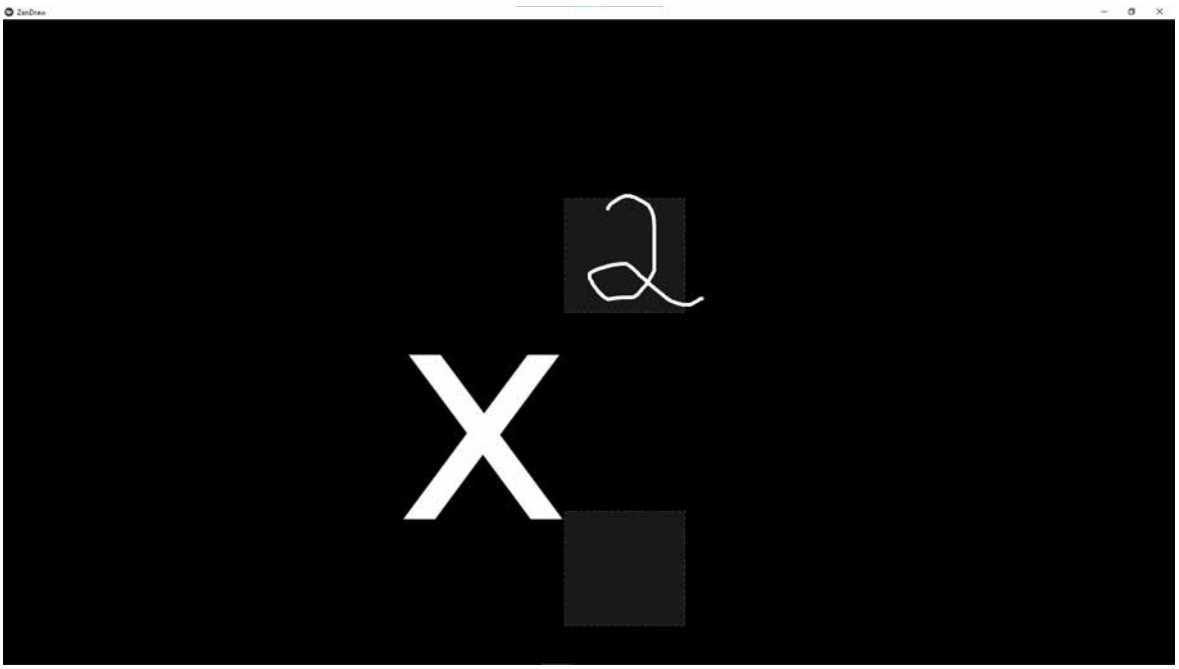

Рисунок 3.9 – Приклад placeholder-зон для приєднання символу

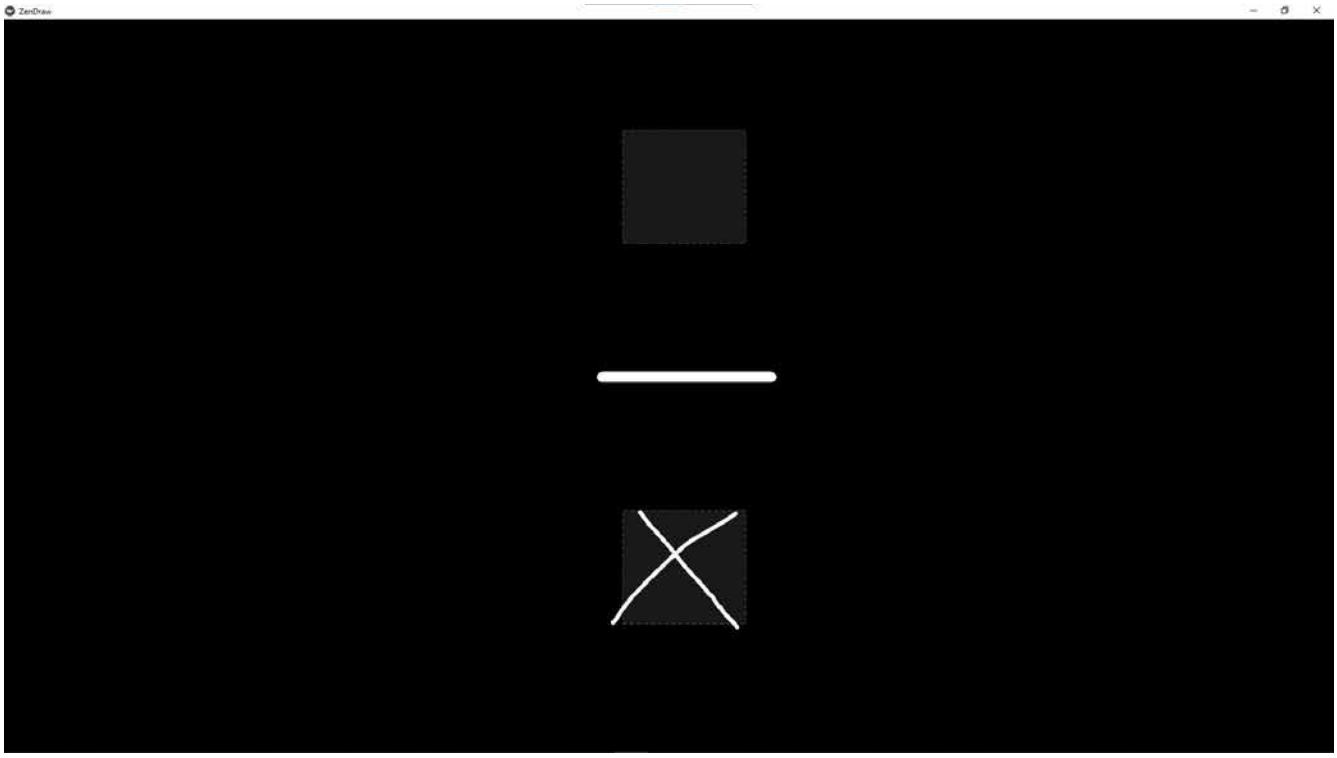

Рисунок 3.10 – Формування дробу засобами LayoutEngine

3.6 Координатні системи, панорамування та масштабування

У Kivy екранна система координат має початок у нижньому лівому куті, а вісь Y спрямована вгору. Натомість LayoutEngine використовує глобальну

систему координат із початком у верхньому лівому куті та віссю Y, спрямованою вниз. Така система зручніша для компонування символів, оскільки схожа на координати прямокутників у більшості графічних бібліотек.

Щоб уникнути помилок позиціонування, у DrawingCanvas реалізовано дві головні функції перетворення: screen_to_world та world_to_screen. Усі операції рендерингу, hit-testing, побудови placeholder-зон і додавання нових символів проходять через ці функції.

Користувач може змінювати масштаб полотна та переміщувати область перегляду. На комп'ютері для цього використовується колесо миші або права кнопка, а на сенсорному пристрої — жести двома пальцями. При масштабуванні важливо, щоб рукописне введення і надруковані символи залишалися в одній координатній системі.

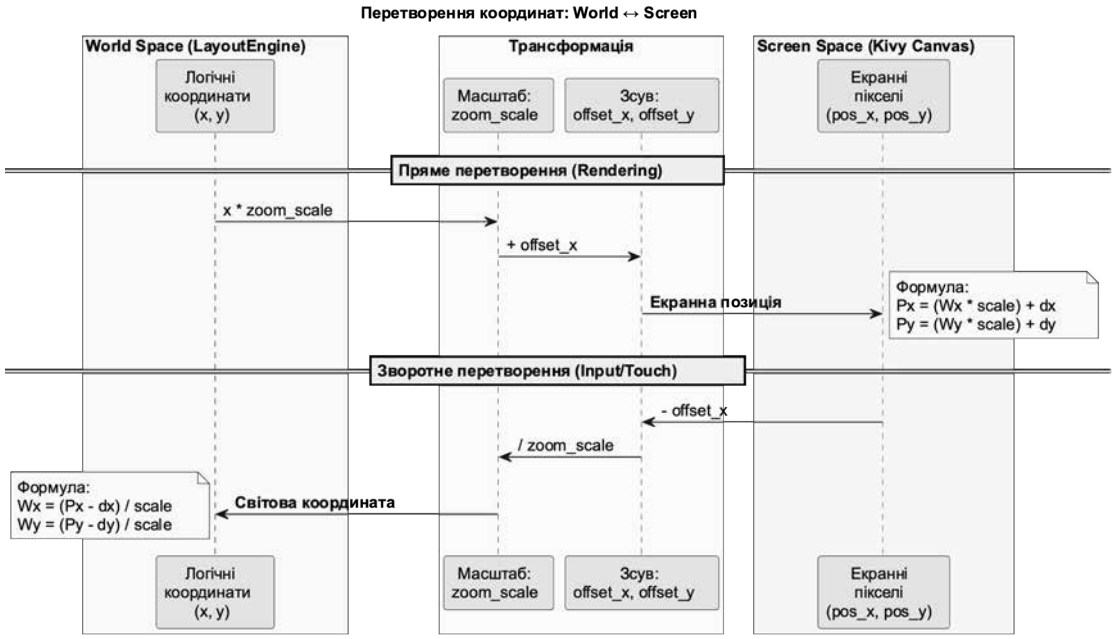

Рисунок 3.11 – Перетворення між екранною та глобальною системами координат

3.7 Редагування, Top-5, збереження та завантаження

Під час рукописного введення неможливо гарантувати, що кожен символ буде розпізнано правильно з першої спроби. Тому у ZenDraw передбачено механізм інтелектуальної корекції. Після розпізнавання символ зберігає список кількох найімовірніших прогнозів, і користувач може перемикатися між ними.

Такий підхід зручніший, ніж повне стирання і повторне малювання символу. Якщо система сплутала символ із близьким за формою, користувач може обрати інший варіант із Top-5. Це зберігає темп роботи та відповідає ідеї природного введення.

Окремо реалізовано збереження та завантаження сесій. Користувач може зберегти поточний математичний вираз у файл JSON, а потім повернутися до нього. Ця функція важлива для навчального використання, коли студент або викладач створює формули поступово.

Функція очищення полотна дозволяє почати роботу спочатку, але перед виконанням такої операції поточний стан може бути збережений у історії. Це дає змогу відновити вираз у разі випадкового очищення.

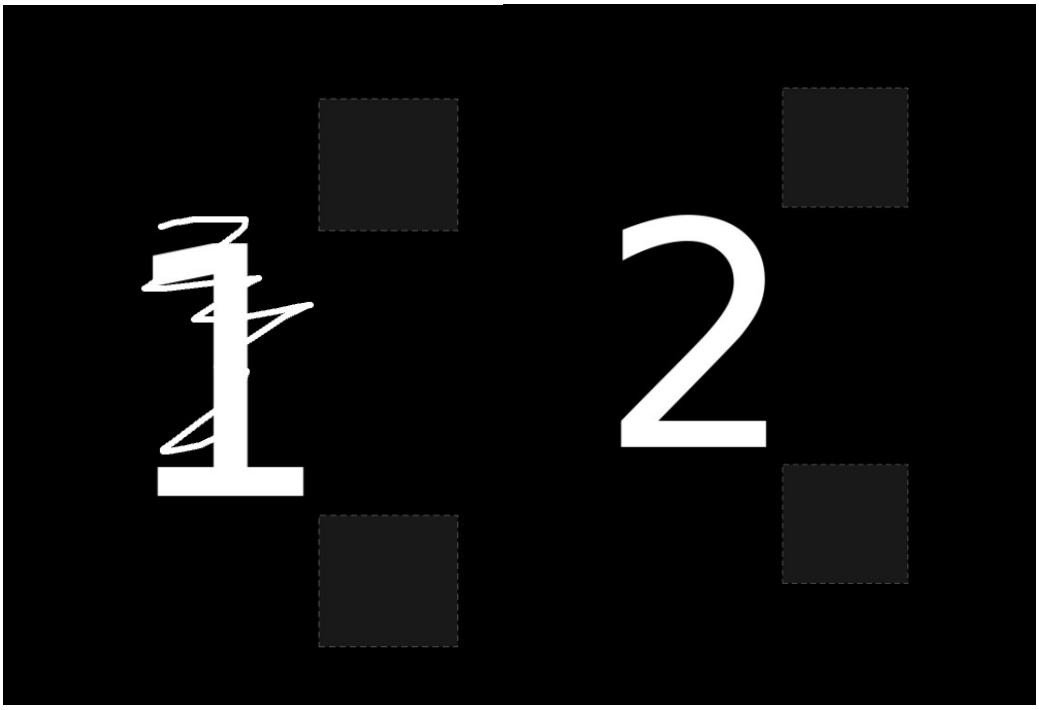

Рисунок 3.12 – Механізм інтелектуальної корекції розпізнаного символу(де з 1 після scrible перетворюється в 2)

Рисунок 3.13 – Збереження сесії користувача

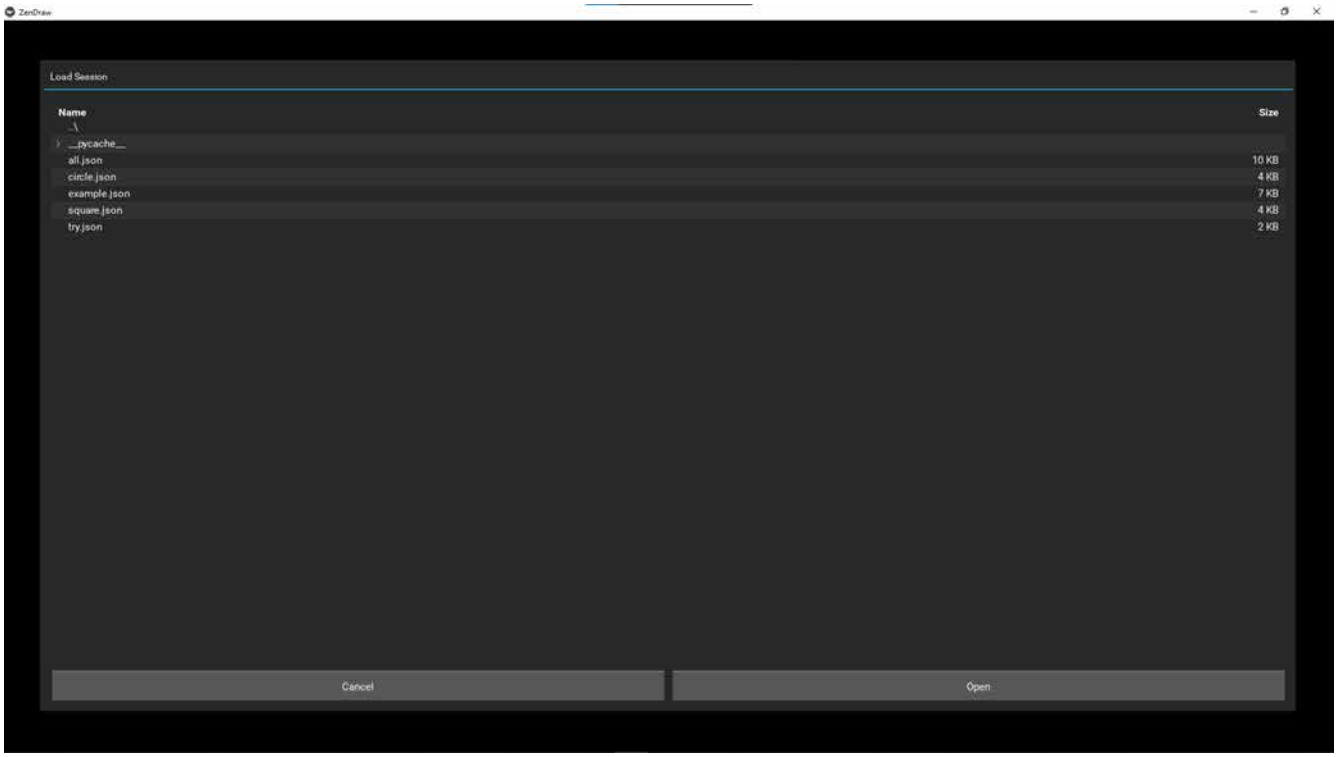

Рисунок 3.14 – Відновлення сесії користувача

3.8 Кросплатформність та особливості мобільної версії

Однією з вимог до ZenDraw є можливість роботи як на комп’ютері, так і на телефоні або планшеті. Для цього було обрано Kivy, оскільки він підтримує

Windows, Linux, macOS та Android. У коді також передбачено перевірку середовища Android, щоб не застосовувати desktop-only налаштування там, де вони можуть завадити реальному сенсорному вводу.

На настільному комп’ютері основними засобами взаємодії є миша, клавіатура і колесо прокрутки. На мобільному пристрої основними засобами є дотики, жести панорамування і масштабування, а також плаваюча кнопка розпізнавання. Тому інтерфейс розроблено так, щоб не залежати від клавіатури.

Відмова від важких ML-залежностей також пов’язана з кросплатформністю. Модель розпізнавання виконується на NumPy, а базові операції із зображенням реалізовано у власному модулі. Це зменшує ризики під час збирання Android-пакета і робить програму більш автономною.

Таблиця 3.4 – Особливості роботи на різних платформах

Платформа	**Основні засоби взаємодії**	**Особливості**
Windows/Linux/macOS	миша, клавіатура, колесо прокрутки	зручно тестувати, редагувати та демонструвати роботу
Android-телефон	палець, FAB-кнопка, жести	застосунок можна використовувати без клавіатури
Android-планшет	стилус або палець, мультитач	найбільш природний сценарій рукописного введення

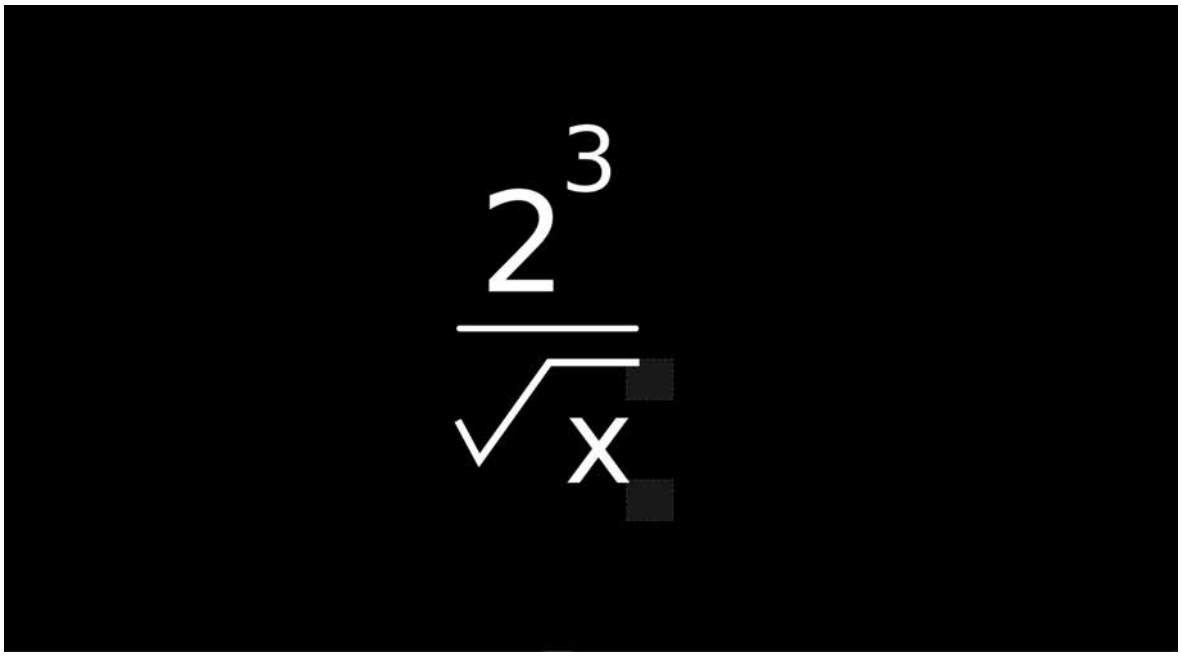

Рисунок 3.15 – Приклад роботи ZenDraw на сенсорному пристрої

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування програмного застосунку

Тестування ZenDraw має охоплювати не лише правильність запуску програми, а й повний сценарій рукописного введення математичного виразу. Оскільки застосунок складається з кількох взаємопов’язаних модулів, тестування доцільно проводити на рівні окремих компонентів і на рівні інтегрованого сценарію.

Модульне тестування застосовується для перевірки геометричних функцій LayoutEngine, обчислення перетину прямокутників, серіалізації вузлів, роботи конфігураційного файла символів, а також допоміжних функцій попередньої обробки зображень. Такі тести не потребують графічного інтерфейсу і можуть виконуватися автоматично.

Інтеграційне тестування перевіряє шлях від рукописного штриха до структурованого символу. Під час такого тесту користувач малює символ, система формує зображення, розпізнає його, додає у LayoutEngine і перемальовує вираз. Особливу увагу потрібно приділяти випадкам, коли символ додається у верхній індекс, знаменник дробу або під корінь.

Функціональне тестування виконується з позиції користувача. Перевіряються сценарії: запуск програми, малювання символу, розпізнавання, виправлення через Top-5, створення степеня, створення дробу, створення кореня, масштабування, панорамування, зміна теми, збереження сесії, завантаження сесії та очищення полотна.

Таблиця 4.1 – Основні тестові сценарії ZenDraw

№	Сценарій	Очікуваний результат
1	Намалювати цифру 2 і запустити розпізнавання	На полотні з’являється друкований символ 2
2	Намалювати x, потім 2 у верхній placeholder-зоні	Система формує вираз x^2
3	Намалювати символ '/', потім чисельник і знаменник	Система формує дріб із горизонтальною рискою
4	Намалювати √ і символ у зоні radicand	Символ розміщується під коренем
5	Перемкнути тему Day/Night	Змінюються кольори фону, символів і допоміжних елементів
6	Виконати undo після додавання символу	Остання дія скасовується
7	Зберегти сесію у JSON і відкрити її повторно	Вираз відновлюється у тому самому вигляді
8	Змінити масштаб полотна	Позиції символів залишаються коректними

Окремим напрямом тестування є перевірка зручності користування. Потрібно оцінити, чи достатньо зрозуміло користувач бачить placeholder-зони, чи не перекривають вони основний вираз, чи не заважає меню роботі з полотном і чи не виникає випадкове приєднання символів до неправильних слотів.

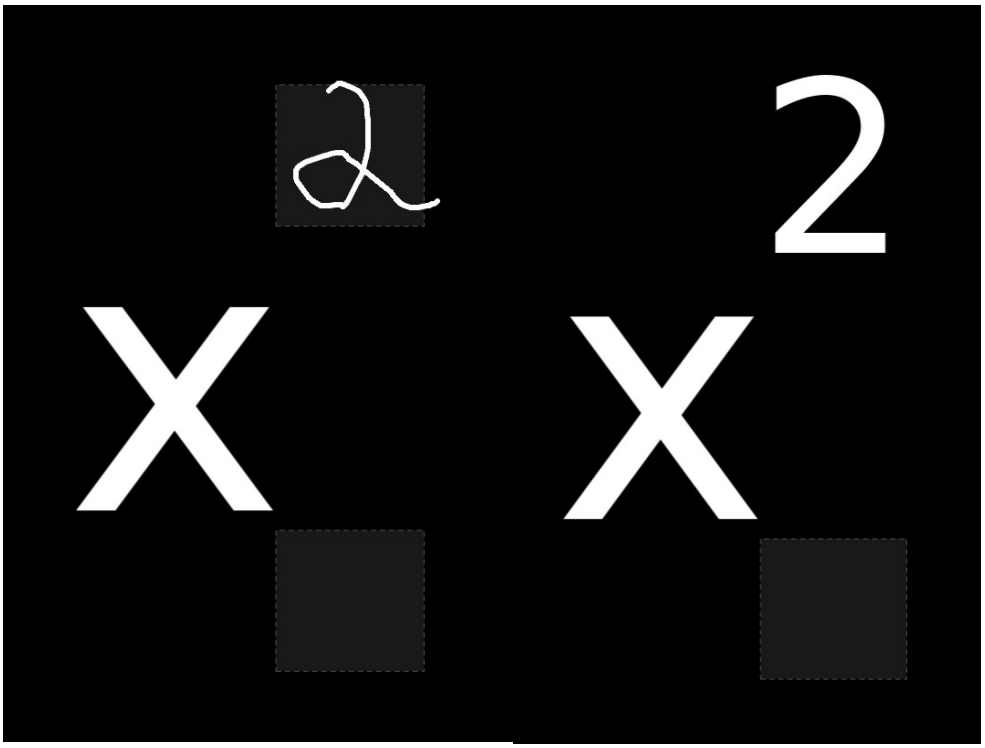

Рисунок 4.1 – Тестування створення верхнього індексу

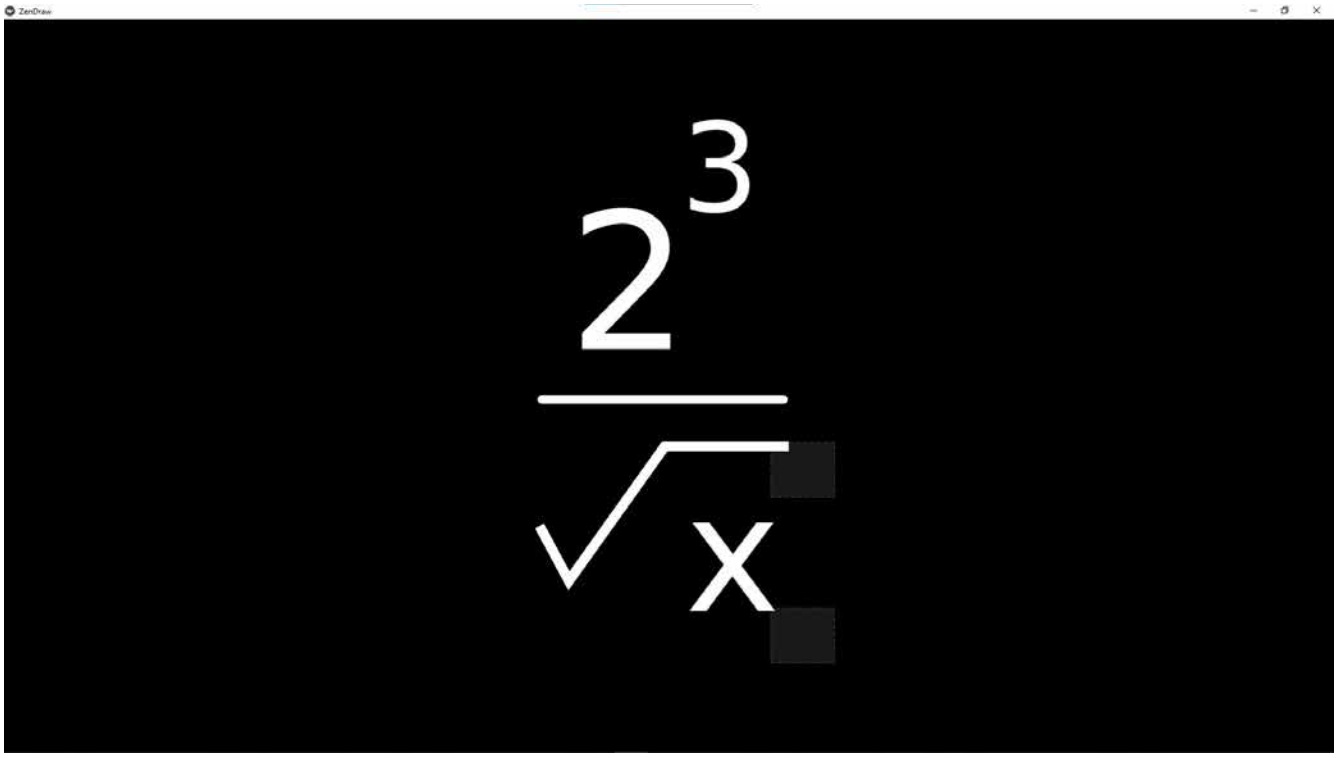

Рисунок 4.2 – Тестування складної математичної структури

4.2 Опис роботи користувача з програмою

Після запуску ZenDraw користувач потрапляє на робоче полотно. Основний принцип роботи полягає в тому, що користувач малює математичний символ у тому місці, де він має з’явитися у виразі. Якщо символ потрібно розмістити як степінь або індекс, його слід намалювати в показаній placeholder-зоні.

Для розпізнавання користувач натискає пробіл на комп’ютері або плаваючу кнопку на сенсорному пристрої. Після цього рукописний штрих замінюється на друкований символ. Якщо результат неправильний, можна скористатися перемиканням між альтернативними прогнозами або повторно намалювати символ.

Меню налаштувань дозволяє змінити тему, перемістити сторону розташування кнопки, зберегти роботу, завантажити сесію, очистити полотно або закрити меню.

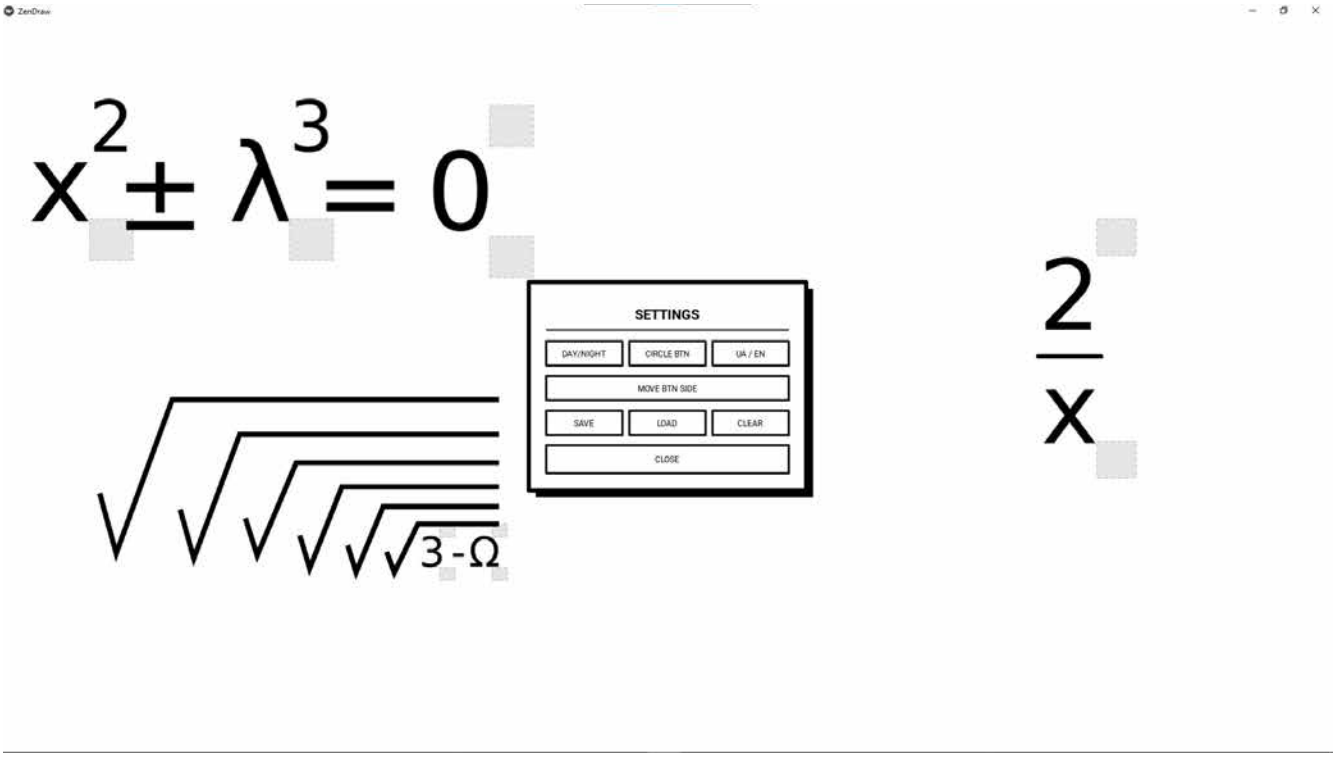

Рисунок 4.3 – Поточний вигляд інтерфейсу ZenDraw із меню налаштувань та placeholder-зонами

На Рис. 4.3 показано, що застосунок може одночасно відображати кілька незалежних або частково пов’язаних математичних структур. У лівій частині розміщено вираз зі степенями, у нижній частині — приклади кореневих конструкцій, а праворуч — дріб. Центральне меню демонструє доступні функції керування.

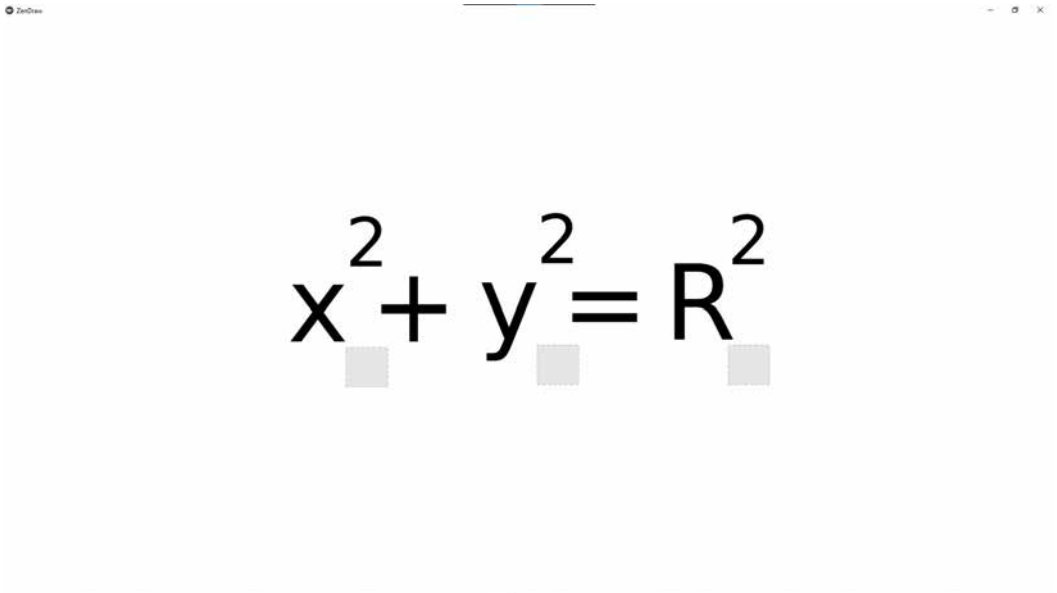

Рисунок 4.4 – Приклад сформованого математичного виразу

4.3 Вимоги до апаратного та програмного забезпечення

Для настільного використання ZenDraw не потребує спеціалізованого обладнання. Базовим варіантом є персональний комп’ютер або ноутбук із встановленим Python-середовищем та необхідними бібліотеками. Для зручнішого рукописного введення можна використовувати графічний планшет або сенсорний екран.

Для мобільного використання бажаним є Android-пристрій із достатнім обсягом оперативної пам’яті та екраном, на якому зручно писати математичні символи. Найкращим сценарієм є планшет зі стилусом, оскільки він найбільш близький до роботи з папером.

Програмні вимоги залежать від способу запуску. Для запуску з вихідного коду потрібні Python, Kivy, NumPy та файли моделі розпізнавання. Для Android-версії потрібне збирання пакета засобами buildozer[18], після чого користувач може запускати застосунок як звичайну мобільну програму.

Таблиця 4.2 – Рекомендовані вимоги до середовища виконання

Компонент	Мінімальні вимоги	Рекомендовані вимоги
Операційна система ПК	Windows 10 / Linux / macOS	Windows 11 або сучасний Linux
Процесор	двоядерний процесор	4 ядра і більше
Оперативна пам’ять	4 ГБ	8 ГБ і більше
Пристрій введення	миша або сенсорний екран	стилус або графічний планшет
Мобільна ОС	Android із підтримкою Kivy-застосунків	Android-планшет зі стилусом
Програмні залежності	Python, Kivy, NumPy	зібраний пакет або віртуальне середовище

Оскільки застосунок працює з локальними файлами сесій, окремий сервер не потрібний. Це спрощує використання у комп’ютерних класах і на особистих пристроях. Водночас для майбутнього розвитку можна передбачити синхронізацію сесій або експорт формул у зовнішні формати.

4.4 Порядок впровадження та експлуатації

Впровадження ZenDraw доцільно виконувати поетапно. На першому етапі перевіряється запуск застосунку на цільовому пристрої. На другому етапі перевіряється робота модуля розпізнавання та наявність файлів ваг моделі. На третьому етапі тестуються основні математичні конструкції. На четвертому етапі користувачам надається коротка інструкція.

Під час експлуатації користувач повинен дотримуватися простого правила: малювати символи в межах видимих зон, якщо потрібно створити структурний зв’язок. Для горизонтального продовження формули символ малюється праворуч від попереднього, для степеня — у верхній зоні, для індексу — у нижній, для дробу — у зоні чисельника або знаменника.

Для збереження результатів роботи рекомендується використовувати окрему папку з JSON-сесіями. Назви файлів бажано робити змістовними, наприклад formula_pythagoras.json або lecture_integrals_01.json. Це полегшує повторне відкриття потрібної роботи.

У разі некоректного розпізнавання користувачеві слід спочатку спробувати перемикання між альтернативними варіантами. Якщо потрібного символу немає серед прогнозів, варто стерти або скасувати дію і намалювати символ ще раз більш чітко.

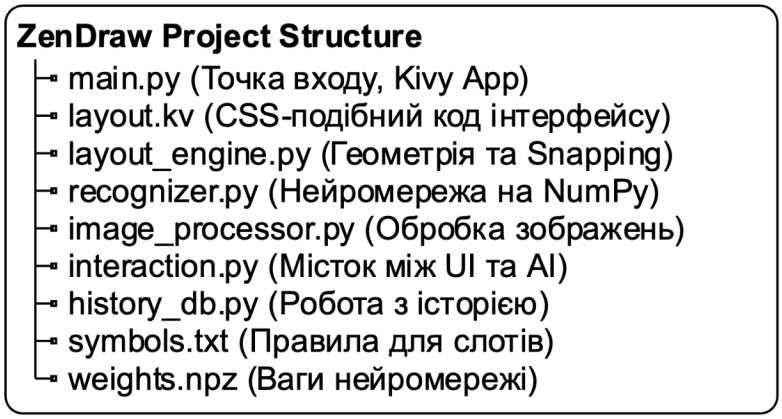

Рисунок 4.5 – Рекомендована структура розгортання застосунку

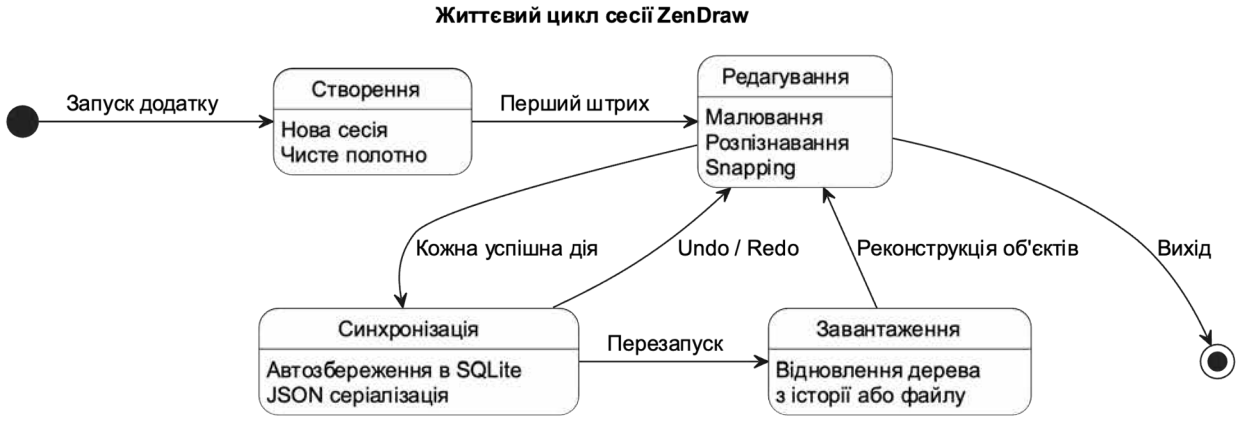

Рисунок 4.6 – Життєвий цикл користувацької сесії ZenDraw

4.5 Обмеження системи та перспективи розвитку

Поточна версія ZenDraw демонструє працездатність основної ідеї рукописного введення математичних виразів, однак має природні обмеження. Якість розпізнавання залежить від навченої моделі, чіткості рукопису та схожості окремих символів. Деякі складні математичні структури можуть потребувати додаткових правил компонування.

Іншим обмеженням є відсутність повноцінного експорту у формати, які можна безпосередньо вставити у наукову роботу, наприклад LaTeX, MathML або SVG. Проте дерево SymbolNode створює добру основу для такого експорту, оскільки структура виразу вже відома.

Перспективним напрямом є покращення моделі розпізнавання за рахунок додаткових навчальних даних, адаптації до конкретного користувача та врахування контексту. Наприклад, якщо символ намальований у степені, система може очікувати цифру, літеру або короткий вираз і відповідно змінювати пріоритети прогнозів.

Також доцільно розвинути інструменти редагування: вибір символу, переміщення піддерева, видалення окремого вузла, групове масштабування, експорт у зображення та автоматичне вирівнювання довгих формул. У мобільній версії варто додати оптимізацію для стилуса та підтримку більш точних жестів.

Таблиця 4.3 – Напрями подальшого розвитку ZenDraw

Напрям	Суть покращення	Очікуваний ефект
Експорт	LaTeX, MathML, SVG або PNG	використання формул у документах і презентаціях
Розпізнавання	донавчання моделі та адаптація до користувача	зменшення кількості помилок
Редагування	видалення вузлів, переміщення піддерев, групове редагування	зручніша робота зі складними виразами
Мобільність	оптимізація для стилуса та планшетів	природніше рукописне введення
Контекст	врахування слота під час вибору прогнозу	краще розпізнавання неоднозначних символів
Навчання	режим тренування символів користувача	персоналізація системи

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі розроблено програмний застосунок ZenDraw, призначений для рукописного введення математичних символів, їх автоматичного розпізнавання та структурованого відображення у вигляді математичних виразів. Розробка спрямована на розв’язання практичної проблеми незручного введення формул у традиційних текстових редакторах і спеціалізованих середовищах.

Під час системного аналізу предметної області визначено, що існуючі засоби введення математичних формул часто потребують знання спеціального синтаксису, великої кількості натискань або роботи з окремими редакторами формул. Запропонований підхід дозволяє наблизити цифрове введення до звичайного письма на папері.

Сформовано функціональні та нефункціональні вимоги до системи. Основними функціями визначено рукописне введення символів, розпізнавання, побудову степенів, індексів, дробів і коренів, підтримку undo/redo, збереження та завантаження сесій, зміну теми і роботу на різних типах пристроїв.

Розроблено інформаційне забезпечення системи. Математичний вираз подано у вигляді дерева вузлів SymbolNode, де кожен символ має координати, масштаб, список прогнозів і набір слотів для дочірніх елементів. Окремо використано конфігураційний файл symbols.txt, який задає відповідність між класами розпізнавання, графічними символами та допустимими слотами.

Реалізовано прикладне програмне забезпечення на основі Python і Kivy. Графічний інтерфейс забезпечує роботу з полотном, рукописним введенням, меню налаштувань, плаваючою кнопкою розпізнавання, жестами масштабування і панорамування. Такий підхід дозволяє використовувати програму як на настільному комп’ютері, так і на телефоні або планшеті.

Реалізовано модуль попередньої обробки зображень image_processor.py, який виконує базові операції без залежності від важких зовнішніх бібліотек. Це підвищує переносимість програми й полегшує збирання мобільної версії.

Реалізовано модуль розпізнавання recognizer.py на основі згорткової нейронної мережі SymbolCNN. Особливістю реалізації є використання NumPy для виконання прямого проходу нейронної мережі без PyTorch або ONNX Runtime під час роботи програми. Це зменшує кількість залежностей і робить застосунок придатнішим для мобільного розгортання.

Розроблено LayoutEngine, який забезпечує структурне компонування математичного виразу. Механізм placeholder-зон і слотів дозволяє користувачу природно формувати степені, нижні індекси, дроби, корені та межі операторів. Новий символ приєднується до структури лише тоді, коли рукописний прямокутник достатньо перекриває відповідну зону.

Проведено опис тестових сценаріїв і рекомендацій щодо впровадження. Визначено вимоги до апаратного та програмного забезпечення, порядок запуску, особливості експлуатації, а також напрями подальшого розвитку. Найважливішими перспективами є експорт у LaTeX/MathML/SVG, покращення моделі розпізнавання, розширення інструментів редагування та оптимізація мобільної версії.

Отже, результатом роботи є функціональний кросплатформний прототип інтелектуального застосунку для природного математичного письма. Практична цінність полягає у можливості використовувати ZenDraw як навчальний, демонстраційний або допоміжний інструмент для швидкого створення математичних виразів без складного клавіатурного введення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LaTeX Project. LaTeX documentation. URL: https://www.latex-project.org/help/documentation/
2. LeCun Y., Bottou L., Bengio Y., Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998. Vol. 86, No. 11. P. 2278-2324. URL: https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition
3. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. URL: https://www.deeplearningbook.org/
4. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2006. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
5. Szeliski R. Computer Vision: Algorithms and Applications. Springer, 2022. URL: https://szeliski.org/Book/
6. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2019. URL: https://www.researchgate.net/publication/365946272_Software_Engineering_A_Practitioner's_Approach_9_th_Edition
7. Sommerville I. Software Engineering. Pearson, 2016. URL: https://dn790001.ca.archive.org/0/items/bme-vik-konyvek/Software%20Engineering%20-%20Ian%20Sommerville.pdf
8. MathML Core Specification. W3C. URL: https://www.w3.org/TR/mathml-core/
9. OpenCV Documentation. Image Processing. URL: https://docs.opencv.org/
10. Harris C. R., Millman K. J., van der Walt S. J. et al. Array programming with NumPy. Nature. 2020. Vol. 585. P. 357-362. URL: https://www.nature.com/articles/s41586-020-2649-2
11. NumPy Developers. NumPy User Guide. URL: https://numpy.org/doc/

12. Kivy. Open source Python framework for rapid development of applications. URL: https://kivy.org/
13. OMG. Unified Modeling Language Specification. URL: https://www.omg.org/spec/UML/
14. SQLite Documentation. URL: https://www.sqlite.org/docs.html
15. Unicode Consortium. The Unicode Standard. URL: https://www.unicode.org/standard/standard.html
16. Pillow Documentation. URL: https://pillow.readthedocs.io/
17. Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/
18. Buildozer Documentation. URL: https://buildozer.readthedocs.io/

ДОДАТОК А

Фрагмент конфігураційного файла symbols.txt

У цьому додатку наведено приклад фрагмента конфігураційного файла, який використовується для визначення доступних математичних символів та їхніх слотів. Повний файл може містити значно більшу кількість класів, проте наведений фрагмент демонструє основний принцип роботи.

```
# Format: id: glyph | slot1 slot2 ...
# Letters and variables
113: x | super sub
114: y | super sub
48: R | super sub
162: λ | super sub
181: Ω | super sub

# Digits
70: 0 | super sub
71: 1 | super sub
72: 2 | super sub
73: 3 | super sub

# Operators and relations
196: + |
195: - |
617: = |
698: ≠ |
601: ≈ |

# Structures
960: √ | radicand
922: / | numer denom

# Large operators
183: ∫ | super sub limit_up limit_down
116: Σ | super sub limit_up limit_down
168: Π | super sub limit_up limit_down
```

Рисунок А.1 – Фрагмент конфігураційного файла

ДОДАТОК Б

Псевдокод основних алгоритмів системи

У додатку наведено псевдокод алгоритмів, які можуть бути використані для пояснення логіки роботи системи під час захисту. Ці алгоритми не є повною копією вихідного коду, але відображають основні кроки, необхідні для розуміння принципу роботи ZenDraw.

```
Input: screen strokes of a handwritten symbol
Output: recognized symbol and top-5 predictions


1. Read stroke coordinates from DrawingCanvas.
2. Calculate bounding rectangle of the ink.
3. Crop the corresponding image area.
4. Convert image to grayscale.
5. Resize image to 32×32 pixels.
6. Normalize pixel values.
7. Pass tensor to SymbolCNN.
8. Apply softmax to network output.
9. Select five classes with highest probabilities.
10. Return the first class as current symbol and save top-5 list.
```

Рисунок Б.1 – Алгоритм розпізнавання рукописного символу

```
Input: recognized glyph, ink rectangle, current expression tree
Output: updated expression tree


1. Create SymbolNode for recognized glyph.
2. Generate all empty slot rectangles for the current expression.
3. For each empty slot:
     calculate overlap_fraction(slot_rect, ink_rect)
4. Select the slot with the highest overlap.
5. If overlap >= SNAP_OVERLAP_THRESHOLD:      attach new node to selected parent and slot
   else:      create new independent root
6. Recalculate layout for the whole expression.
7. Save state to history.
8. Redraw canvas.
```

Рисунок Б.2 – Алгоритм приєднання символу до математичного виразу

```
Input: user action undo or redo
Output: restored expression state


1. If user performs a change:
     serialize current expression to JSON
     push state to history
2. If undo is requested:
     move history pointer one step back
     deserialize previous JSON state
     redraw expression
3. If redo is requested:
     move history pointer one step forward
     deserialize next JSON state
     redraw expression
4. If no previous/next state exists:
     keep current expression unchanged.
```

Рисунок Б.3 – Алгоритм скасування та повторення дій

ДОДАТОК В

Додаткові скріншоти інтерфейсу

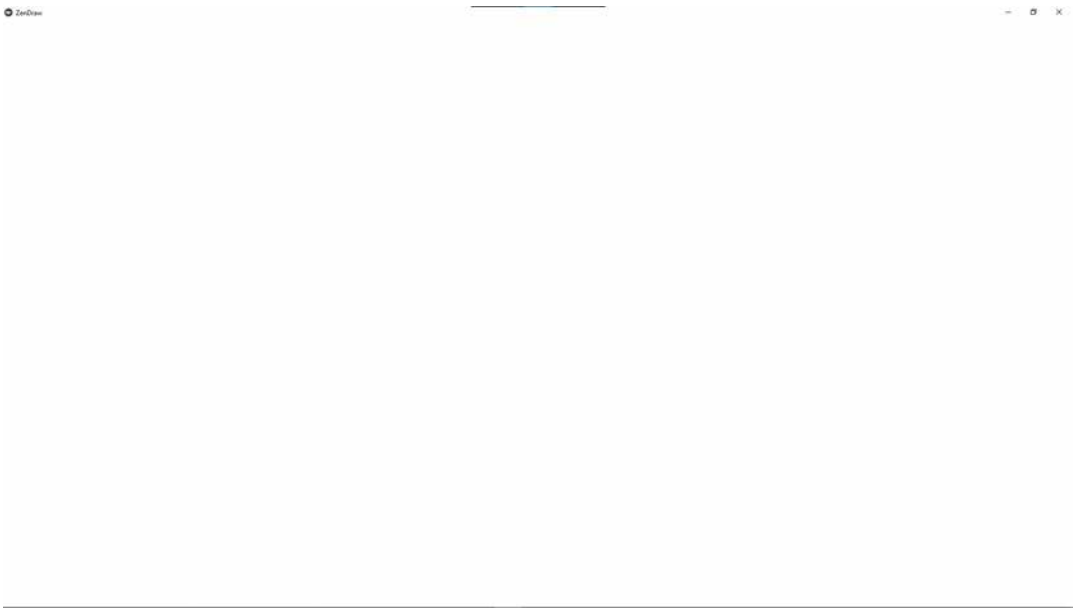

Рисунок В.1 – Порожнє полотно після запуску програми.

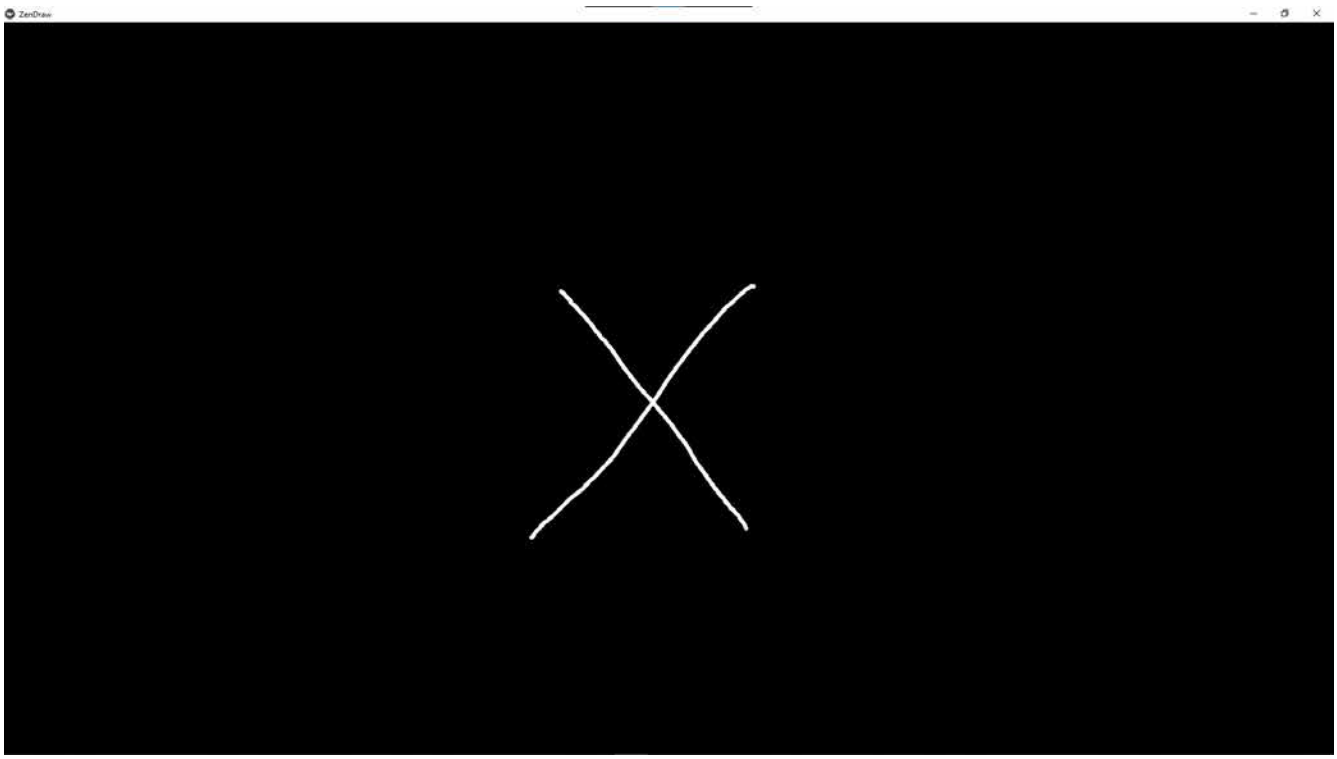

Рисунок В.2 – Рукописне введення символу перед розпізнаванням.

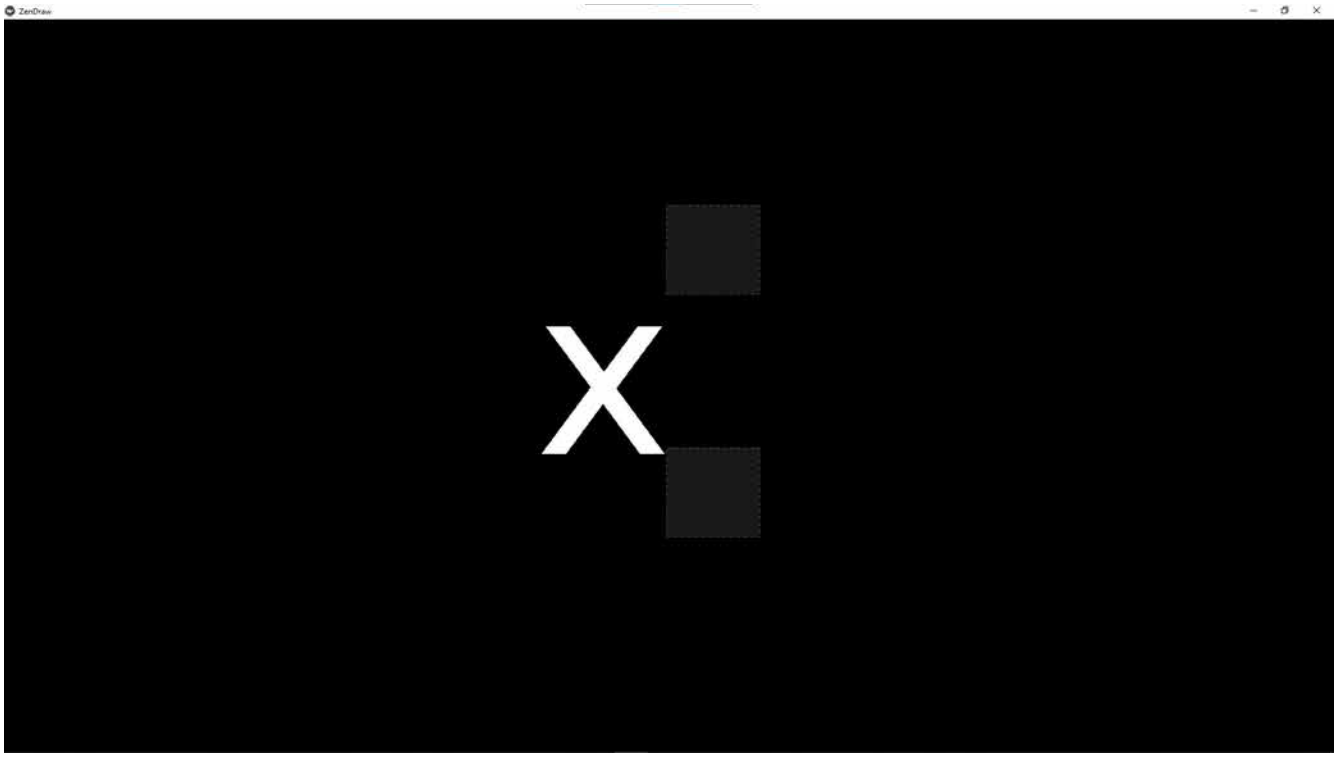

Рисунок В.3 – Результат розпізнавання одного символу.

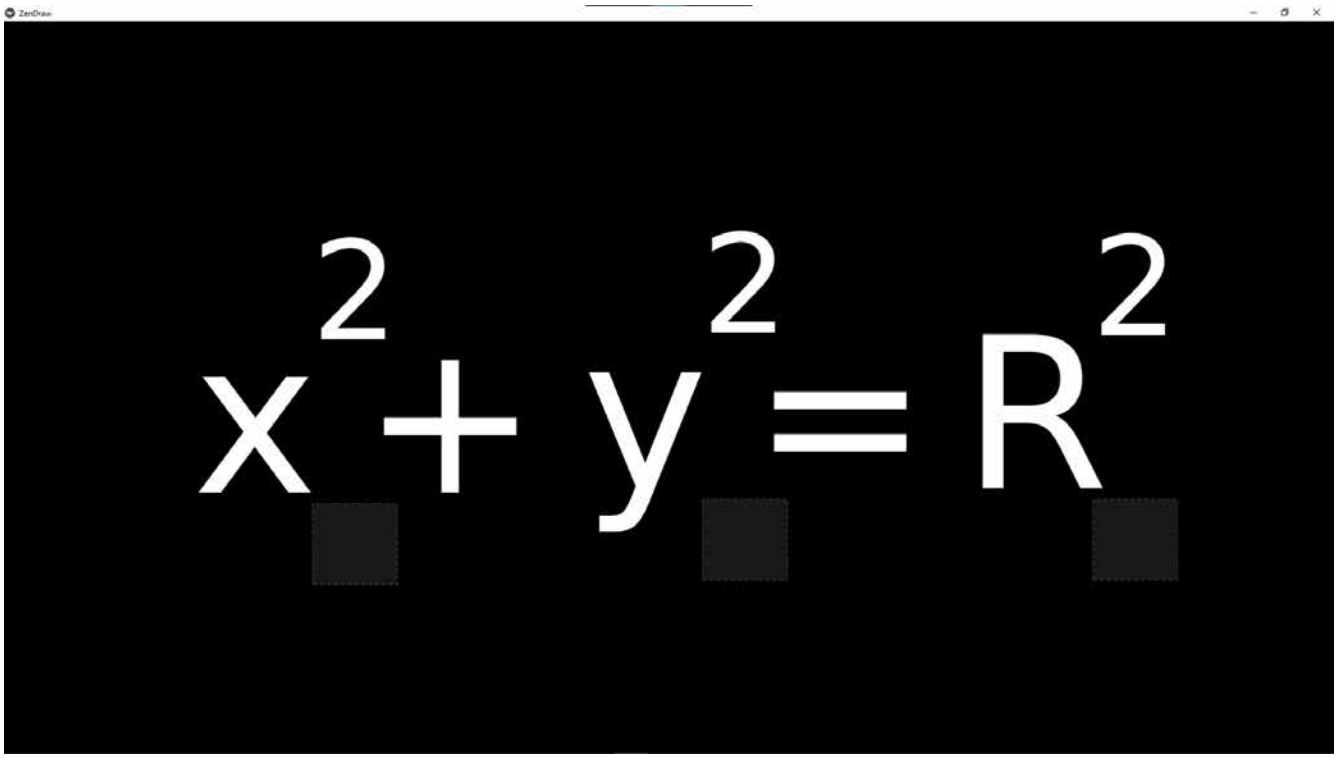

Рисунок В.4 – Створення виразу $x^2 + y^2 = R^2$.

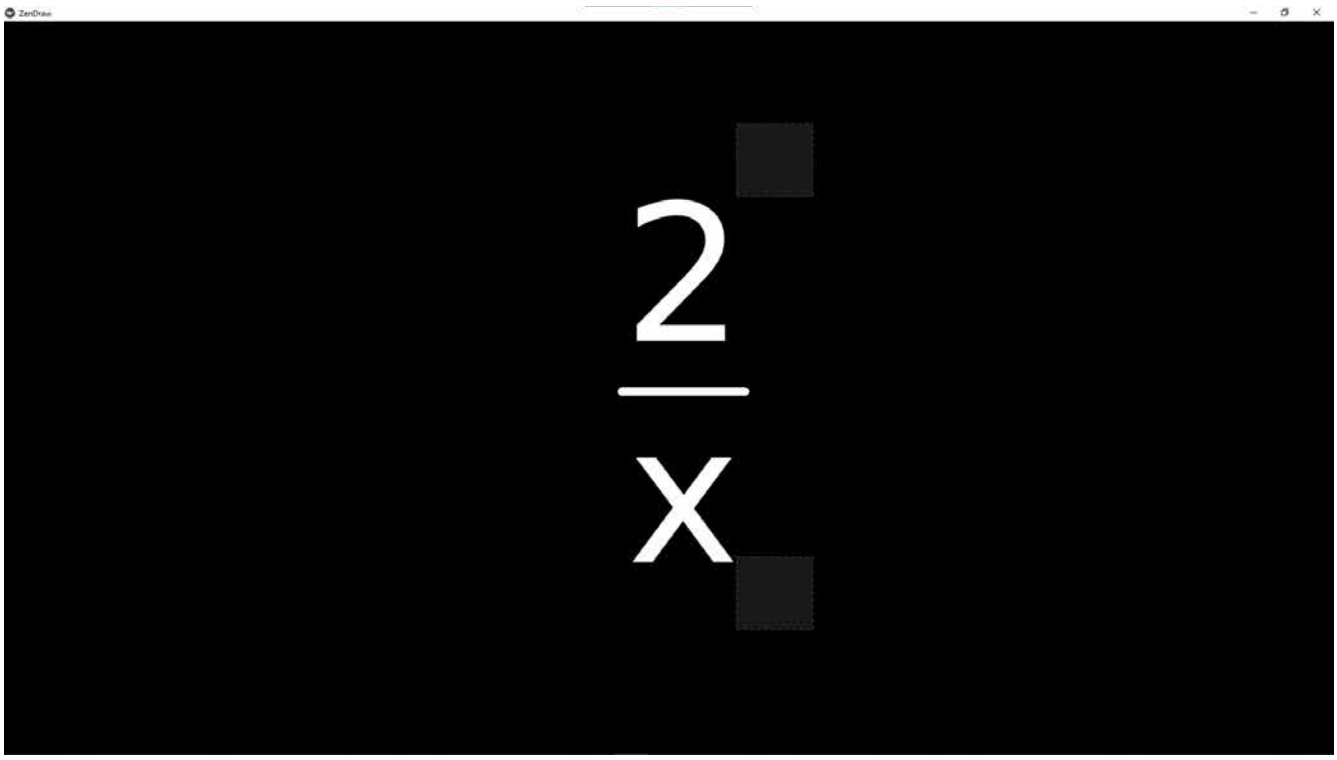

Рисунок В.5 – Створення дробу 2/x.

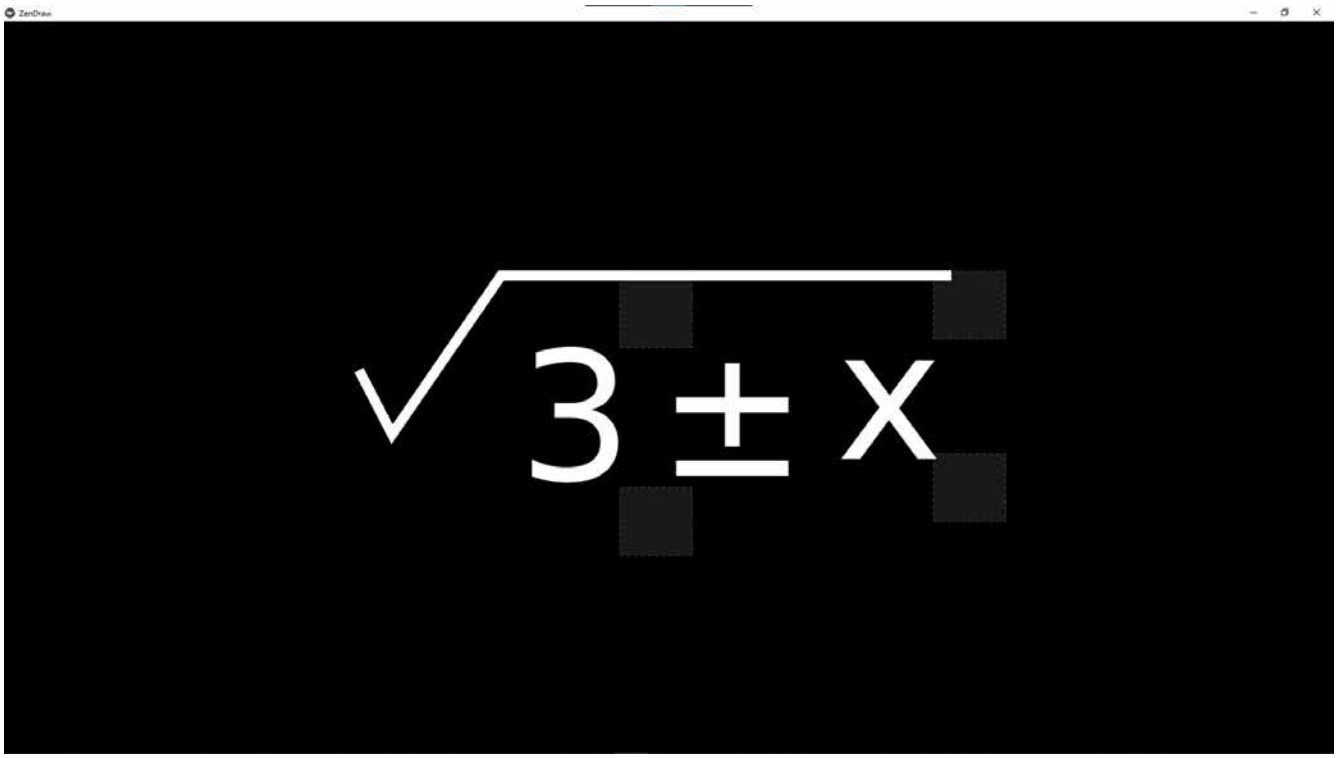

Рисунок В.6 – Створення кореня з підкореневим виразом.

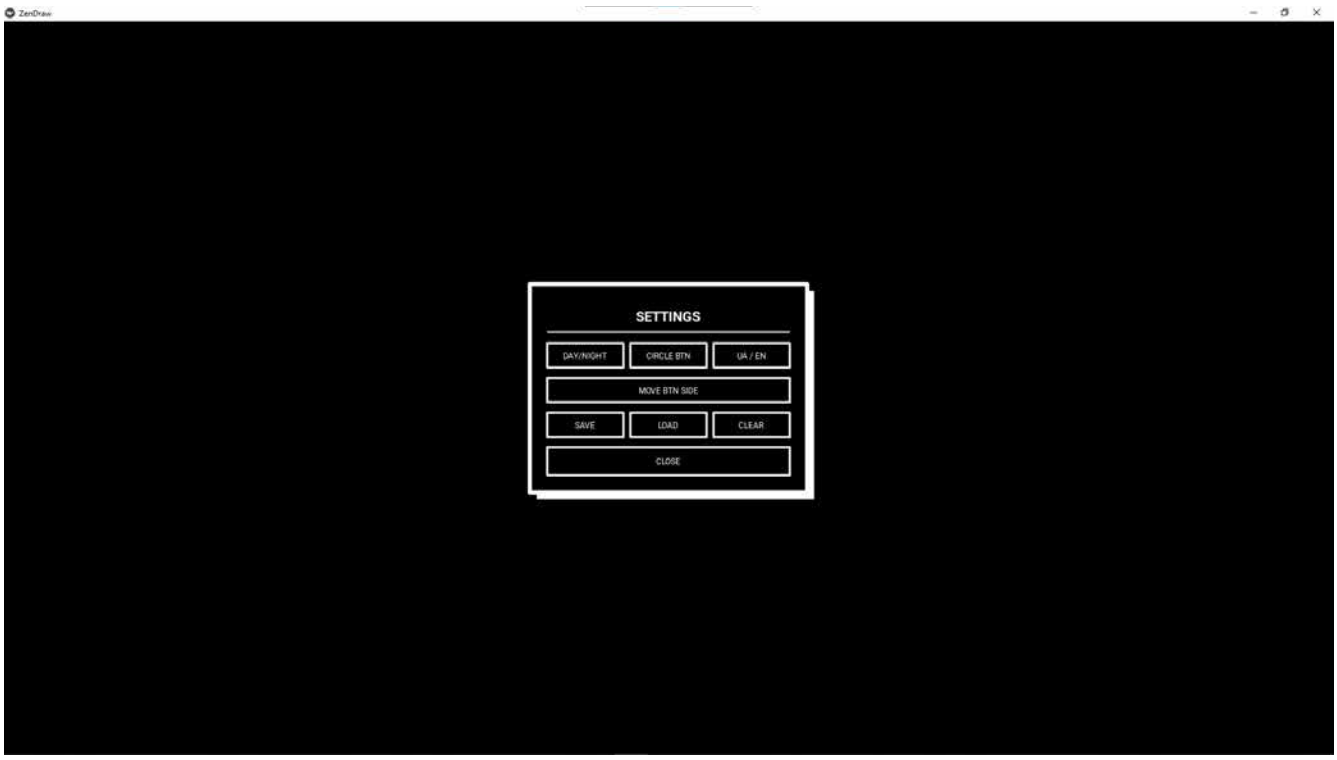

Рисунок В.7 – Перемикання Day/Night mode.

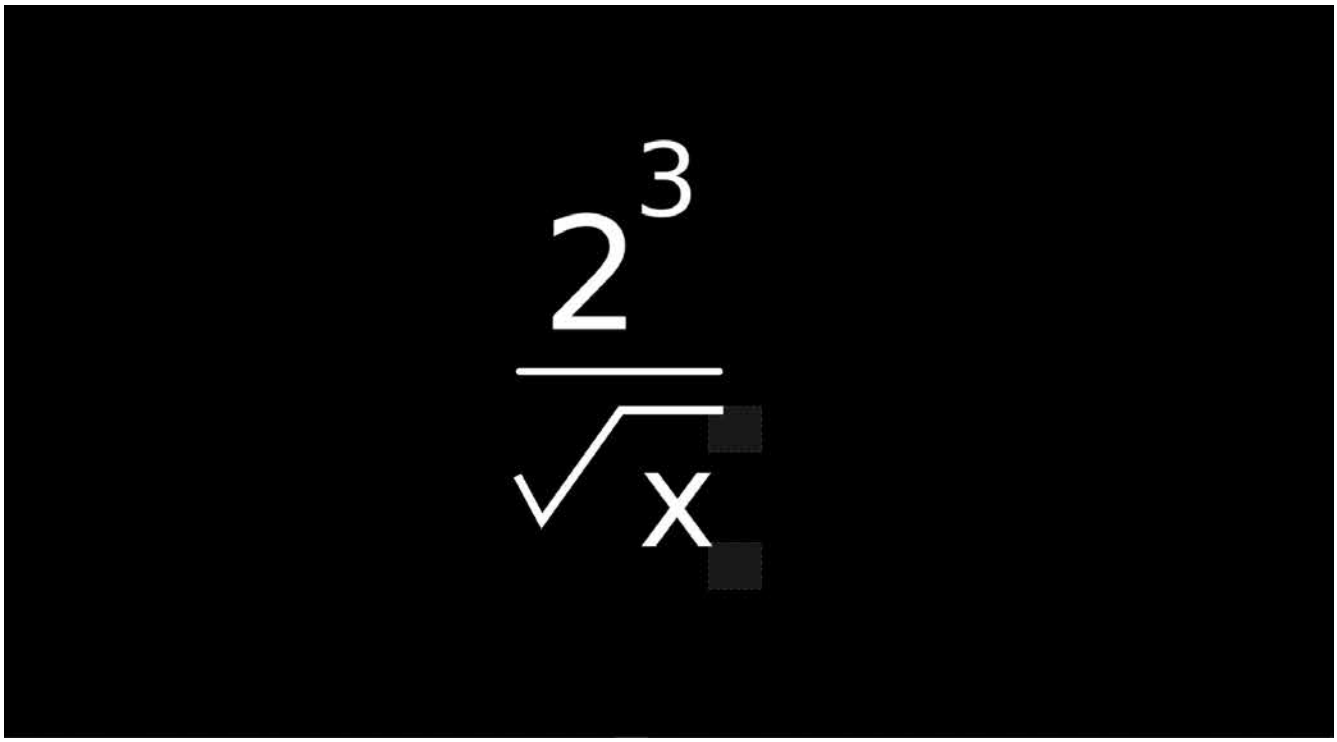

Рисунок В.8 – Запуск або тестування на телефоні/планшеті.

ДОДАТОК Ж

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Очеретянко Владислав Володимирович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

- Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Система розпізнавання рукописного тексту з використанням методів комп'ютерного зору» є результатом моєї особистої праці.
- Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
- Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

    1. [ ] інструменти генеративного ШІ не використовувалися.
    2. [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        1. [ ] перекладу іншомовних джерел;
        2. [+] стилістичного редагування та перевірки граматики;
        3. [+] допомоги у написанні/відлагодженні програмного коду;
        4. [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Владислав Очеретянко**
(підпис)