Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Веселова Вячеслава Вячеславовича*
(ПІБ)

академічної групи *122-19-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для мобільних пристроїв з метою аналізу економічних показників діяльності підприємства*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Швачич Г.Г.*			
розділів:				
спеціальний	*проф. Швачич Г.Г.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-19-2* *Веселова Вячеслава Вячеславовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для мобільних пристроїв з метою аналізу економічних показників діяльності підприємства*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *проф. Швачич Г.Г.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Веселов В.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 100 с., 47 рис., 1 табл., 3 дод., 20 джерел.

Об’єкт розробки: програмне забезпечення для мобільних пристроїв для аналізу економічних показників діяльності підприємства.

Мета кваліфікаційної роботи: створення програмного забезпечення для мобільних пристроїв, яке містить функціонал для розрахунку економічних показників діяльності підприємства та амортизації, для скорочення витрат часу на здійснення обчислень економічних показників, а також для надання зручності виконання розрахунків.

У вступі наведено мету роботи та галузь застосування, розглянуто актуальність розробки, а також описано постановку завдання.

У першому розділі здійснено аналіз предметної галузі, описано призначення розробки, вказано галузь застосування мобільного додатка, сформульовано постановку завдання, наведено вимоги до застосунку, його функціональних характеристик, а також до параметрів технічних засобів для використання створеного додатка.

У другому розділі описано функціональне призначення мобільного додатка, наведено відомості про використані мови програмування та технології для розробки додатка, описана його структура та алгоритм функціонування, визначено організацію вхідних та вихідних даних, охарактеризовано використані програмні та технічні засоби для створення застосунку, розглянуто виклик та завантаження додатка, а також проведено детальний опис графічного інтерфейсу додатка.

В економічному розділі виконано розрахунки трудомісткості створеного мобільного додатка, а також розраховано вартість роботи по розробці застосунку та витрати на його створення.

Практичне значення полягає у створенні мобільного додатка, який надає можливість зручного виконання обчислень амортизації та економічних показників підприємства, а також є мобільним рішенням, яке можна використовувати на пристроях з різними операційними системами без необхідності підключення до мережі Інтернет.

Актуальність розробки полягає у необхідності обчислення та аналізу економічних показників підприємства для його успішної роботи, а також у відсутності мобільного крос-платформного рішення для здійснення таких розрахунків без підключення до мережі Інтернет.

Список ключових слів: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МОБІЛЬНІ ПРИСТРОЇ, ПІДПРИЄМСТВО, ЕКОНОМІЧНІ ПОКАЗНИКИ, ОБЧИСЛЕННЯ, МОБІЛЬНИЙ ДОДАТОК.

ABSTRACT

Explanatory note: 100 p., 47 fig., 1 table, 3 app., 20 sources.

Object of development: software for mobile devices for analyzing enterprise economic indicators.

Purpose of the qualification work: creation of software for mobile devices that contains functionality for calculating economic indicators of an enterprise and amortization to reduce the time spent on calculating economic indicators and to provide convenience in performing calculations.

In the introduction, the purpose of the work and the field of application are presented, the relevance of the development is considered, and the task is described.

The first chapter analyzes the subject area, describes the purpose of the development, indicates the field of usage of the mobile application, formulates the task statement, provides requirements for the application, its functional characteristics, and the parameters of technical devices for using the created application.

The second chapter describes the functional purpose of the mobile application, provides information on the programming languages and technologies used to develop the application, describes its structure and functioning algorithm, defines the organization of input and output data, characterizes the software and hardware used to create the application, describes the launch of the application and downloading, and provides a detailed description of the application's graphical interface.

The economic chapter contains calculations of the labor intensity of the created mobile application, as well as the cost of the application development work and the cost of its creation.

The practical significance is to create a mobile application that provides an opportunity to conveniently calculate amortization and economic indicators of an enterprise and is a mobile solution that can be used on devices with different operating systems without an Internet connection.

The relevance of the development lies in the need to calculate and analyze the economic indicators of an enterprise for its success, as well as in the lack of a mobile, cross-platform solution for making such calculations without an Internet connection.

List of keywords: SOFTWARE, MOBILE DEVICES, ENTERPRISE, ECONOMIC INDICATORS, CALCULATIONS, MOBILE APPLICATION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕОМ - електронно-обчислювальна машина;

ОС - операційні системи;

ПЗ - програмне забезпечення;

API - Application Programming Interface;

APK - Android Package;

CSS - Cascading Style Sheets;

EXE - Executable File;

HTML - HyperText Markup Language;

IOS - iPhone Operating System;

MacOS - Macintosh Operating System.

ВСТУП

Розробка програмного забезпечення є частиною сфери діяльності фахівця спеціальності Комп’ютерні науки. Створення мобільних додатків, тобто ПЗ для мобільних пристроїв, в свою чергу, є досить важливою складовою цього напряму.

Метою кваліфікаційної роботи є створення програмного забезпечення для мобільних пристроїв, яке містить функціонал для розрахунку економічних показників діяльності підприємства та обчислення амортизації, для скорочення витрат часу на здійснення обчислень амортизації та розрахунку економічних показників, а також для надання зручності виконання розрахунків.

Розроблене ПЗ призначене для використання у сфері економіки підприємства.

Актуальність розробки даного програмного забезпечення полягає у необхідності обчислення та аналізу економічних показників діяльності підприємства для його успішної роботи в сучасному світі, а також у відсутності такого програмного забезпечення для мобільних пристроїв з різними операційними системами для здійснення таких обчислень без підключення до мережі Інтернет.

До основних задач, які треба виконати для того, щоб досягнути мету даної роботи, належать наступні:

- здійснити аналіз предметної галузі та наявних рішень проблеми;
- сформулювати постановку завдання та вимоги до ПЗ;
- розробити алгоритм роботи мобільного додатку, враховуючи такі нештатні ситуації, як введення користувачем некоректного значення;
- здійснити реалізацію мобільного додатку та тестування ПЗ.

Відповідно до здійсненого аналізу розроблене програмне забезпечення має задовольняти таким основним функціональним можливостям:

– зменшення витрат часу на виконання розрахунків економічних показників діяльності підприємства та амортизації;

– зниження ризику допускання помилки при здійсненні обчислень;

– забезпечення зручності виконання обчислень;

– забезпечення роботи ПЗ на пристроях з різними ОС, а також без необхідності підключення до мережі Інтернет.

Загалом, програмне забезпечення значно прискорює та полегшує виконання обчислень амортизації та економічних показників підприємства, реалізуючи при цьому зручне здійснення розрахунків, а також забезпечуючи функціонування без необхідності підключення до мережі Інтернет на пристроях, які працюють на різних ОС.

Розроблений мобільний додаток може бути використаний економістами, які працюють в сфері економіки підприємства. Крім того, дане програмне забезпечення може бути використано спеціалістами в області інформаційних технологій для вдосконалення вже розробленого застосунку з метою використання для інших підприємств та організацій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Галуззю застосування розроблюваного програмного забезпечення є галузь економіки підприємства. Економіка підприємства – це галузь економічної науки, яка заснована на використанні економічних законів, а також певних закономірностей роботи та розвитку виробництва [1].

Предмет вивчення економіки підприємства – способи та методи ефективного використання елементів процесу виробництва на рівні підприємства, а також їх раціонального поєднання [2, с. 10].

Для забезпечення ефективної діяльності підприємства досить важливим є обчислення економічних показників роботи підприємства, а також їх аналіз.

Наразі не існує розробленого ПЗ для виконання розрахунків економічних показників діяльності підприємства та здійснення обчислень амортизації, яке призначене для використання на мобільних пристроях, а також може функціонувати на різних ОС та без необхідності підключення до мережі Інтернет.

Найближчим подібним програмним забезпеченням, яке містить функціонал для розрахунку амортизації, є веб-сервіс «Нарахування амортизації основних засобів в бухгалтерському обліку» [3].

За допомогою цього веб-сервісу можна обчислити амортизаційні відрахування прямолінійним методом, методом зменшення залишкової вартості, прискореного зменшення залишкової вартості, кумулятивним та виробничим методами. Його графічний інтерфейс представлений на рис. 1.1.

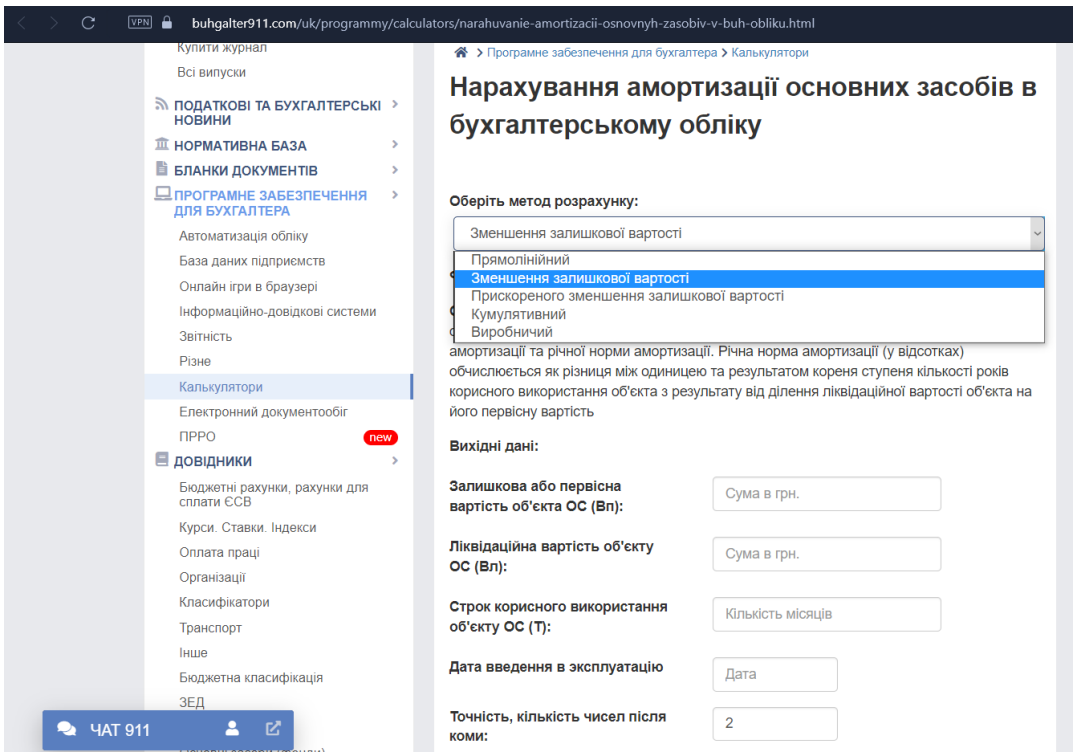

Рис. 1.1. Графічний інтерфейс веб-сервісу

Приклад розрахунку амортизації методом зменшення залишкової вартості наведений на рис. 1.2.

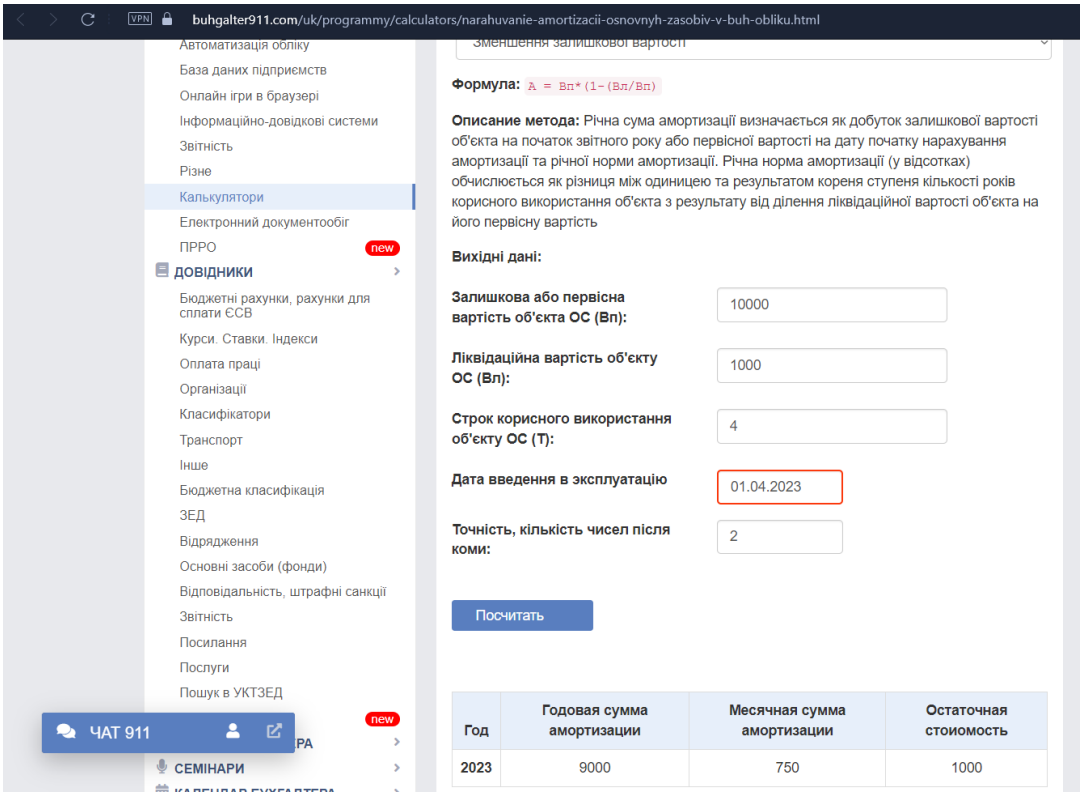

Рис. 1.2. Розрахунок амортизації за допомогою веб-сервісу

Приведений веб-сервіс дозволяє розрахувати лише амортизацію, але він не надає можливість розрахувати економічні показники підприємства. Крім того, для використання даного сервісу необхідне постійне підключення до мережі Інтернет, отже за його відсутності веб-сервіс не зможе функціонувати.

1.2. Призначення розробки та галузь застосування

Програмне забезпечення, що розроблюється, має назву «Мобільний додаток для аналізу економічних показників діяльності підприємства».

Призначення розроблюваного мобільного додатку полягає в наданні користувачам функціоналу з метою виконання обчислень амортизації та економічних показників підприємства, який є зручним у використанні та доступним для використання на мобільних пристроях під управлінням різних ОС, що дозволяє значно скоротити витрати часу на виконання розрахунків та зменшити ймовірність допускання помилки в розрахунках.

До основних причин виникнення необхідності розробки даного додатку можна віднести такі:

- виконання обчислень амортизації та економічних показників займає багато часу;

- при проведенні розрахунків має місце значна ймовірність допустити помилку, причому при збільшенні об’єму обчислень, ймовірність помилки зростає;

- здійснення розрахунків без ПЗ не є достатньо зручним;

- відсутність ПЗ із заданим функціоналом для використання на мобільних девайсах.

Основною цільовою галуззю, в якій може бути використаний розроблюваний додаток, є галузь економіки підприємства.

1.3. Підстави для розробки

Розробка даного дипломного проекту на тему «Розробка програмного забезпечення для мобільних пристроїв з метою аналізу економічних показників діяльності підприємства» ведеться на підставі наказу по Національному технічному університету «Дніпровська політехніка» від 16.05.2023р. № 350-с.

1.4. Постановка завдання

Метою створення розроблюваного мобільного додатку є скоротити обсяг витрат часу на здійснення розрахунків амортизації та економічних показників підприємства, зменшити показник імовірності допускання помилки в обчисленнях, а також надати зручний у використанні інтерфейс для виконання обчислень, не використовуючи при цьому підключення до мережі Інтернет.

Призначення даного ПЗ полягає в наданні функціоналу для виконання розрахунків економічних показників діяльності підприємства та амортизації, доступного для використання на мобільних девайсах, які використовують різні операційні системи.

Сутність завдання полягає в розробці програмного забезпечення для мобільних пристроїв, яке дозволяє обчислювати економічні показники підприємства та амортизацію, скорочуючи при цьому витрати часу на здійснення розрахунків. Рішення даного завдання є доцільним, оскільки розрахунок амортизації та економічних показників є важливим процесом для організації ефективної роботи підприємства, а здійснення цих обчислень без ПЗ, яке призначене для цього, призводить до великих витрат часу.

Вихідною інформацією, яку надає мобільний застосунок, є результати обчислень відповідно до обраного користувачем економічного показника, що необхідно обчислити, або методу розрахунку амортизації, якщо обчислюється амортизація. Ця інформація призначена для подальшого аналізу та використання користувачем.

Вхідна інформація вводиться користувачем у текстові поля. Крім того, всі дані, введені користувачем, перевіряються на відповідність вимогам до цих даних. Якщо значення, яке вводить користувач, не є числом або не належить до діапазону допустимих значень, то виводиться відповідне повідомлення.

Здійснення обчислень припиняється в тому випадку, якщо користувач вводить значення, що не належить до діапазону допустимих значень, або не вводить значення взагалі, залишаючи текстове поле порожнім. В цих випадках виводиться діалогове вікно з повідомленням про некоректність введеного значення або про те, що текстове поле є порожнім, відповідно. Після цього користувачу надається можливість ввести коректне значення. В інших випадках припинення виконання обчислень не передбачається.

Слід зазначити, що даний мобільний додаток призначений для вирішення таких задач, як обчислення економічних показників підприємства та здійснення розрахунку амортизації, а також не має зв’язку з будь-якими іншими задачами.

Розподілу функцій між персоналом та технічними засобами при вирішенні завдань системи не відбувається, оскільки всі обчислення виконуються самим мобільним додатком, а користувач лише обирає те, що саме необхідно розрахувати.

1.5. Вимоги до програми або програмного виробу

В даному підрозділі проводиться опис функцій, що мають бути реалізовані в програмному забезпеченні, що розроблюється. Крім того, приводиться опис вхідних та вихідних даних, вимоги до технічних характеристик пристроїв для забезпечення функціонування ПЗ, а також вимоги до мови програмування та програмних засобів, що використовуються для створення та забезпечення роботи мобільного додатку.

1.5.1. Вимоги до функціональних характеристик

До функцій, які необхідно реалізувати у розроблюваному додатку, належать наступні функції:

– перегляд списку можливих операцій та здійснення вибору операції, яку необхідно виконати;

– надання можливості введення даних та здійснення їх перевірки, а також виведення діалогового вікна при введенні значень, що не задовольняють вимогам до вхідних даних;

– обчислення амортизації за допомогою прямолінійного методу, методу зменшення залишкової вартості, методу прискореного зменшення залишкової вартості, а також кумулятивного методу;

– розрахунок економічних показників (середньорічна вартість основних фондів, коефіцієнт вибуття, коефіцієнт відновлення, коефіцієнт зносу, номінальний та дійсний фонди, коефіцієнти екстенсивного та інтенсивного використання, фондовіддача, фондоємність, фондоозброєність, інтегральний коефіцієнт використання, а також показник рентабельності основних фондів);

– надання можливості округлити отриманий результат здійснених розрахунків та вказати кількість знаків після коми, до якої необхідно здійснити округлення;

– відображення результату виконаних обчислень.

Дані мають вводитися користувачем у відповідні текстові поля. Крім того, повинна здійснюватися перевірка введених значень на відповідність до вимог до даних, що вводяться. Це необхідно для того, щоб забезпечити коректне функціонування ПЗ та отримати правильний результат обчислень.

Результати здійснених розрахунків повинні виводитись під текстовими полями для введення даних та кнопки для виконання обчислень на відповідних екранах мобільного застосунку.

1.5.2. Вимоги до інформаційної безпеки

Варто зазначити, що питання інформаційної безпеки є досить важливими для даного програмного забезпечення. В даному випадку вимоги до інформаційної безпеки забезпечуються вимогами до безпеки мобільних пристроїв, на яких встановлюється та використовується даний мобільний додаток.

1.5.3. Вимоги до складу та параметрів технічних засобів

Розроблене програмне забезпечення орієнтоване для використання на мобільних пристроях, оскільки вони можуть використовуватись будь-де. Крім того, дане ПЗ може використовуватись і для десктопних пристроїв. Адже на сьогоднішній день користувачі частіше використовують мобільні пристрої ніж настільні [4].

Додаток призначений для функціонування на девайсах, які повинні працювати на мобільних операційних системах Android та IOS, оскільки вони є найбільш розповсюдженими серед мобільних платформ [5].

Оскільки алгоритми виконання обчислень прописуються в коді самого ПЗ, то мобільні пристрої, для яких розроблюється дане ПЗ, не потребують підключення до мережі Інтернет.

Інших специфічних вимог до технічних характеристик мобільних девайсів для користування цим додатком не встановлюється, оскільки даний додаток не використовує багато системних ресурсів, а також не вимагає потужного пристрою для використання.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для створення, а також подальшого функціонування розробленого мобільного додатку, мова програмування, яка використовується для розробки

додатку, повинна забезпечувати можливість розробки мобільних додатків для операційних систем Android та IOS, оскільки саме ці ОС є цільовими для використання розроблюваного застосунку.

Крім того, має місце використання крос-платформного фреймворку, який забезпечує ефективніший процес розробки додатків для мобільних пристроїв. При цьому створені додатки будуть функціонувати однаково на тих ОС, які підтримуються даним фреймворком. Крім того, крос-платформна розробка мобільних додатків дозволить скоротити витрати часу та коштів, а також зробити процес розробки та підтримки простішим [6].

Оскільки розроблюване ПЗ призначається для виконання обчислень відповідно до даних, введеним користувачем, а також для відображення результату, то немає потреби у використанні бази даних для розробки та функціонування цього мобільного додатку.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Функціональним призначенням мобільного додатку визначаються наступні функції:

– здійснення розрахунку амортизації з використанням прямолінійного методу, методу зменшення залишкової вартості, методу прискореного зменшення залишкової вартості та кумулятивного методу;

– обчислення основних економічних показників. До них належать такі показники: середньорічна вартість основних фондів, коефіцієнт вибуття, коефіцієнт відновлення, коефіцієнт зносу, номінальний та дійсний фонди, коефіцієнти екстенсивного та інтенсивного використання, фондовіддача, фондоємність, фондоозброєність, інтегральний коефіцієнт використання, показник рентабельності основних фондів.

При цьому ПЗ надає функцію округлення результату, отриманого внаслідок виконання обчислень, до вказаної користувачем кількості знаків після коми.

Для дослідження основних методів амортизації було використано [7 - 9].

Експлуатаційне призначення розробленого додатку полягає в забезпеченні скорочення витрат часу на виконання обчислень амортизації та обчислення економічних показників підприємства, зниження ймовірності допускання помилки при здійсненні розрахунків, а також в наданні зручного інтерфейсу для здійснення обчислень.

2.2. Опис застосованих математичних методів

Для розробленого ПЗ є важливим використання математичних методів. В основному в процесі розробки використовувались економіко-математичні

моделі для визначення та аналізу основних економічних показників діяльності підприємства.

До основних моделей, які використовувались в даній кваліфікаційній роботі, належать наступні:

- модель (2.1), яка використовувалась для визначення повної первісної вартості;

- модель (2.2), яка була використана для обчислення коефіцієнтів вибуття та відновлення;

- модель (2.3), яка використовувалась для обчислення номінального фонду;

- модель (2.4), яка була використана для визначення дійсного фонду;

- модель (2.5), яка використовувалась для визначення фондовіддачі;

- модель (2.6), яка використовувалась для обчислення фондоємності;

- модель (2.7), яка була використана для визначення фондоозброєності.

$$Ф_П = З_{ПР} + З_{ТР} + З_М + З_{БМР}, \text{ грн}, \quad (2.1)$$

де $З_{ПР}$ – витрати на придбання устаткування, грн,

$З_{ТР}$ – витрати на транспортування, грн,

$З_М$ – витрати на установку, грн,

$З_{БМР}$ – витрати на будівельні роботи (для споруд), грн.

$$K_В = \frac{Ф_{ВИБ}}{Ф_{ПР}}, \qquad K_{ОН} = \frac{Ф_{ВВ}}{Ф_{КР}}, \quad (2.2)$$

де $Ф_{ВИБ}$ - вартість вибулих фондів, грн,

$Ф_{ПР}$ - вартість основних фондів на початок року, грн,

$Ф_{ВВ}$ - вартість введених фондів, грн,

$Ф_{КР}$ - вартість основних фондів на кінець року, грн.

$$F_Н = (Д_К - Д_{СВ} - Д_{ВИХ}) \cdot S \cdot T_{ЗМ}, \text{ годин}, \quad (2.3)$$

де $Д_К, Д_{СВ}, Д_{ВИХ}$ – число календарних, святкових та вихідних днів у році відповідно,

S – кількість змін на добу,

$T_{ЗМ}$ – тривалість зміни, годин.

$$F_Д = F_H \left(1 - \frac{\alpha}{100}\right), \text{ годин,} \tag{2.4}$$

де α - частка часу на планово-попереджувальні ремонти, %.

$$Ф_O = \frac{Q}{Ф_{СР}}, \text{ грн/грн,} \tag{2.5}$$

де Q – річний випуск продукції, грн,

$Ф_{СР}$ – середньорічна вартість, грн.

$$Ф_E = \frac{Ф_{СР}}{Q} = \frac{1}{Ф_O}, \text{ грн/грн,} \tag{2.6}$$

де $Ф_O$ – фондовіддача.

$$Ф_{ОЗ} = \frac{Ф_{СР}}{Ч_{ПВП}}, \text{ грн/чол,} \tag{2.7}$$

де $Ч_{ПВП}$ – чисельність промислово-виробничого персоналу, чол.

2.3. Опис використаних технологій та мов програмування

Для використання в процесі розробки мобільного додатку було вирішено використовувати мову програмування Python. Ця мова є потужною, має досить високу швидкодію, може бути легко інтегрована з іншими технологіями, легка у вивченні, а також має відкритий вихідний код та є безкоштовною у використанні [10].

Слід зазначити, що основними факторами, через які програмісти обирають Python, є наступні: якість ПЗ, продуктивність роботи розробника, переносимість програм, бібліотеки підтримки, а також інтеграція компонентів [11, с. 4].

Крім того, для розробки програмного забезпечення для мобільних

пристроїв Python вдало поєднується з фреймворком Kivy, який призначений для розробки мобільних додатків для девайсів, що використовують різні операційні системи, а також значно полегшує та пришвидшує процес розробки, оскільки надає велику кількість корисних компонентів для використання при створенні застосунку. Цим пояснюється доцільність використання даних технологій для розробки крос-платформного мобільного додатку.

Kivy – це фреймворк із відкритим вихідним кодом, який використовується для створення крос-платформних додатків. Цей фреймворк є безкоштовним та легким у використанні, а також він надає можливість розробляти додатки для таких операційних систем: Android, iOS, macOS, Windows та Linux [12]. Kivy приваблює велику кількість розробників через такі ключові причини: Kivy має вбудовану підтримку Multitouch-девайсів, є єдиним способом писати код мовою Python для мобільних пристроїв, замінює жахливі API старіших графічних інтерфейсів на кшталт HTML та CSS, а також забезпечує підтримку одного додатку для великої кількості ОС [13, с. 1].

2.4. Опис структури системи та алгоритмів її функціонування

Мобільний додаток складається з 18 екранів, один з яких є головним та відкривається при запуску додатку. З цього екрану можна перейти на інші екрани, які призначені для виконання розрахунків амортизації або обчислення економічного показника, обраного користувачем. Крім того, з кожного екрану для виконання обчислень можна повернутись на головний екран.

Блок-схема алгоритму роботи екрану, призначеного для обчислення амортизації прямолінійним методом, представлена на рис. 2.1. Реалізація даного алгоритму наведена в лістингу 2.1.

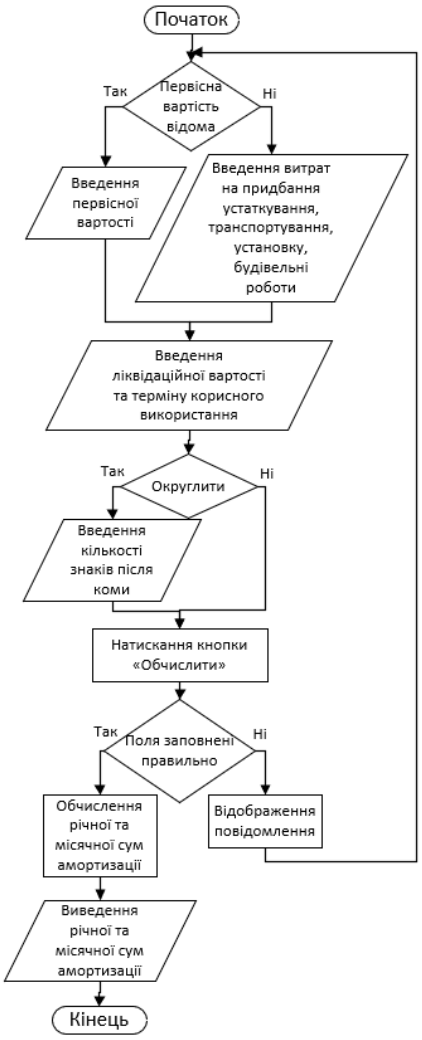

Рис. 2.1. Блок-схема алгоритму роботи екрану розрахунку амортизації прямолінійним методом

Лістинг 2.1

Реалізація алгоритму екрану обчислення амортизації прямолінійним методом

```
# Метод, що розраховує амортизацію прямолінійним методом
    def calculate(self):
        # Якщо опція "Первісна вартість відома" обрана
        if self.ids.initial_value_is_known_switch.active:
            # Отримання введених користувачем даних з полів для введення
            initial_value_input = self.initial_value_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
```

```
        elif self.values_start_with('0', [initial_value_input, liquidation_value_input, term_of_useful_use_input]):
            self.show_dialog("Значення мають бути більше за нуль!")
        # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [initial_value_input, liquidation_value_input]):
            self.show_dialog("Значення не може починатись з крапки!")
        else:
            # Перетворення значень з полів для введення з рядків на числа
            initial_value = float(initial_value_input)
            liquidation_value = float(liquidation_value_input)
            # Обчислення та відображення результату
            self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)
    else: # якщо опція "Первісна вартість відома" не обрана
        # Отримання введених користувачем даних з полів для введення
        equipment_purchase_costs_input = self.equipment_purchase_costs_text_field.text
        transportation_costs_input = self.transportation_costs_text_field.text
        installation_costs_input = self.installation_costs_text_field.text
        construction_costs_input = self.construction_costs_text_field.text
        liquidation_value_input = self.ids.liquidation_value.text
        term_of_useful_use_input = self.ids.term_of_useful_use.text

        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([
            equipment_purchase_costs_input,
            transportation_costs_input,
            installation_costs_input,
            construction_costs_input,
            liquidation_value_input,
            term_of_useful_use_input
        ]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
        elif self.values_start_with('-', [
            equipment_purchase_costs_input,
            transportation_costs_input,
            installation_costs_input,
            construction_costs_input,
            liquidation_value_input,
            term_of_useful_use_input
        ]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [
            equipment_purchase_costs_input,
            transportation_costs_input,
            installation_costs_input,
            construction_costs_input,
            liquidation_value_input
        ]):
            self.show_dialog("Значення не може починатись з крапки!")
        # Якщо значення витрат на придбання устаткування починається з '0', то відобразити діалогове вікно
        elif equipment_purchase_costs_input[0] == '0':
            self.show_dialog("Значення витрат на придбання устаткування має бути більше за нуль!")
        # Якщо значення ліквідаційної вартості починається з '0', то відобразити діалогове вікно
        elif liquidation_value_input[0] == '0':
            self.show_dialog("Значення ліквідаційної вартості має бути більше за нуль!")
        # Якщо значення терміну корисного використання починається з '0', то відобразити діалогове вікно
        elif term_of_useful_use_input[0] == '0':
            self.show_dialog("Значення терміну корисного використання має бути більше за нуль!")
        else:
```

Продовж. лістингу 2.1

```
            # Обчислення первісної вартості та перетворення значень з полів для введення з рядків на числа
            initial_value = float(equipment_purchase_costs_input) + float(transportation_costs_input) +
float(installation_costs_input) + float(construction_costs_input)
            liquidation_value = float(liquidation_value_input)
            # Обчислення та відображення результату
            self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)
```

Блок-схема алгоритму роботи екрану, призначеного для розрахунку амортизації методом зменшення залишкової вартості, представлена на рис. 2.2. Реалізація алгоритму зображена в лістингу 2.2.

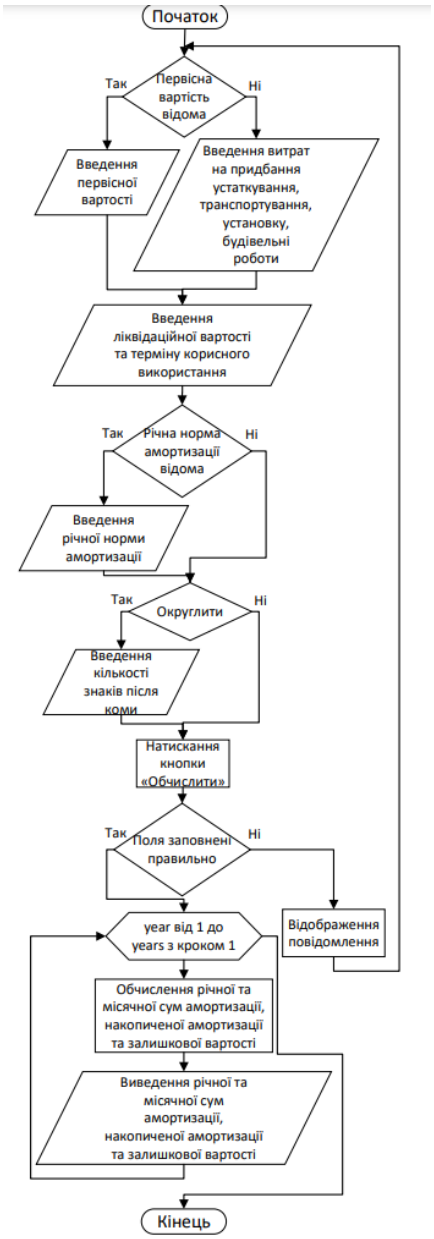

Рис. 2.2. Блок-схема алгоритму роботи екрану розрахунку амортизації методом зменшення залишкової вартості

Лістинг 2.2

Реалізація алгоритму екрану обчислення амортизації методом зменшення залишкової вартості

```
# Метод, що розраховує амортизацію методом зменшення залишкової вартості
    def calculate(self):
        # Якщо опція "Первісна вартість відома" обрана
        if self.ids.initial_value_is_known_switch.active:
            # Отримання введених користувачем даних з полів для введення
            initial_value_input = self.initial_value_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
            elif self.values_start_with('0', [initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Значення мають бути більше за нуль!")
            # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то відобразити
діалогове вікно
            elif self.values_start_with('.', [initial_value_input, liquidation_value_input]):
                self.show_dialog("Значення не може починатись з крапки!")
            else:
                # Перетворення значення первісної вартості з рядка на число
                initial_value = float(initial_value_input)
                # Обчислення в залежності від річної норми амортизації
                self.calculate_according_to_annual_depreciation_rate(initial_value, liquidation_value_input,
term_of_useful_use_input)
        else: # якщо опція "Первісна вартість відома" не обрана
            # Отримання введених користувачем даних з полів для введення
            equipment_purchase_costs_input = self.equipment_purchase_costs_text_field.text
            transportation_costs_input = self.transportation_costs_text_field.text
            installation_costs_input = self.installation_costs_text_field.text
            construction_costs_input = self.construction_costs_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input,
                term_of_useful_use_input
            ]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input,
```

```
            term_of_useful_use_input
        ]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [
            equipment_purchase_costs_input,
            transportation_costs_input,
            installation_costs_input,
            construction_costs_input,
            liquidation_value_input
        ]):
            self.show_dialog("Значення не може починатись з крапки!")
        # Якщо значення витрат на придбання устаткування починається з '0', то відобразити діалогове вікно
        elif equipment_purchase_costs_input[0] == '0':
            self.show_dialog("Значення витрат на придбання устаткування має бути більше за нуль!")
        # Якщо значення ліквідаційної вартості починається з '0', то відобразити діалогове вікно
        elif liquidation_value_input[0] == '0':
            self.show_dialog("Значення ліквідаційної вартості має бути більше за нуль!")
        # Якщо значення терміну корисного використання починається з '0', то відобразити діалогове вікно
        elif term_of_useful_use_input[0] == '0':
            self.show_dialog("Значення терміну корисного використання має бути більше за нуль!")
        else:
            # Обчислення первісної вартості та перетворення значень з полів для введення з рядків на числа
            initial_value = float(equipment_purchase_costs_input) + float(transportation_costs_input) + float(installation_costs_input) + float(construction_costs_input)
            # Обчислення в залежності від річної норми амортизації
            self.calculate_according_to_annual_depreciation_rate(initial_value, liquidation_value_input, term_of_useful_use_input)
```

Реалізація алгоритмів роботи інших екранів наводиться в Додатку А.

Блок-схема алгоритму роботи екрану, призначеного для обчислення амортизації методом прискореного зменшення залишкової вартості, представлена на рис. 2.3.

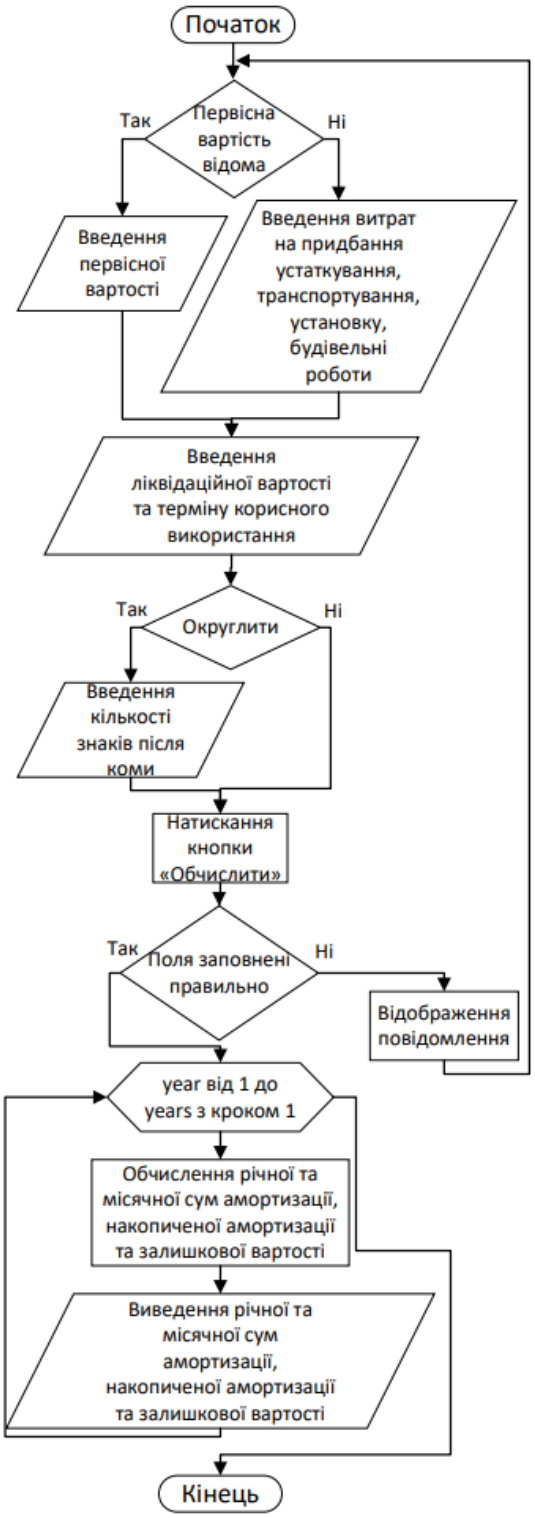

Рис. 2.3. Блок-схема алгоритму роботи екрану розрахунку амортизації методом прискореного зменшення залишкової вартості

Блок-схема алгоритму роботи екрану, призначеного для обчислення амортизації кумулятивним методом, представлена на рис. 2.4.

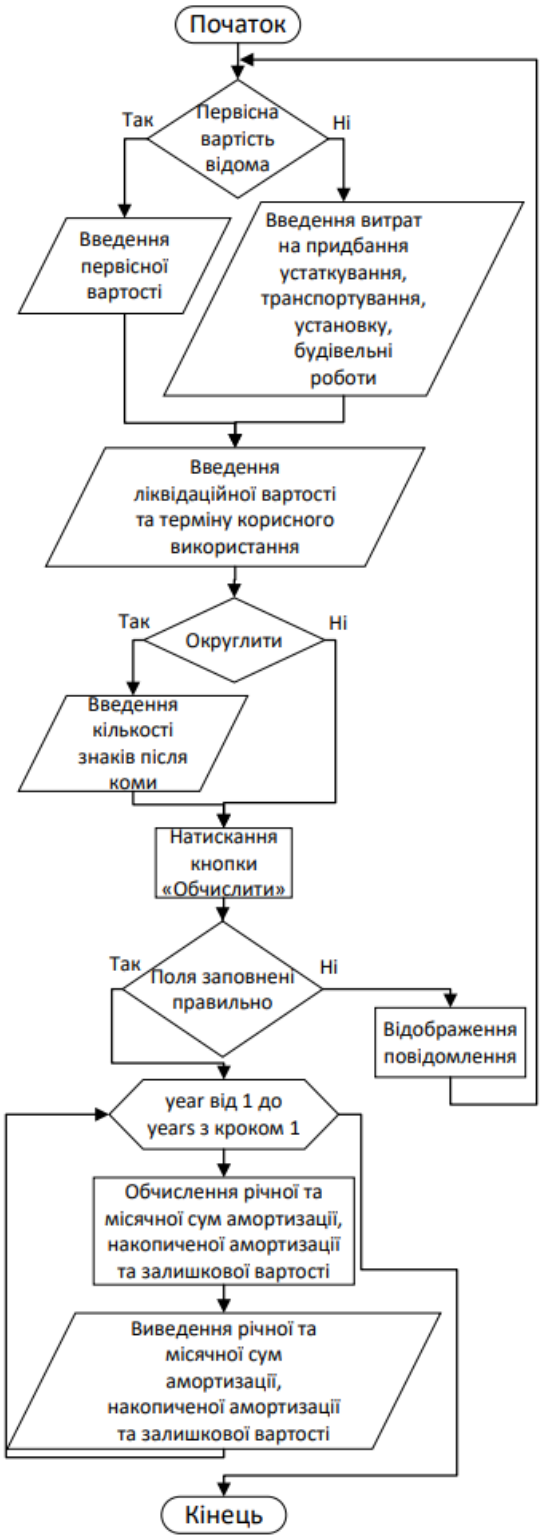

Рис. 2.4. Блок-схема алгоритму роботи екрану розрахунку амортизації кумулятивним методом

Блок-схема алгоритму роботи екрану, призначеного для розрахунку середньорічної вартості основних фондів, представлена на рис. 2.5.

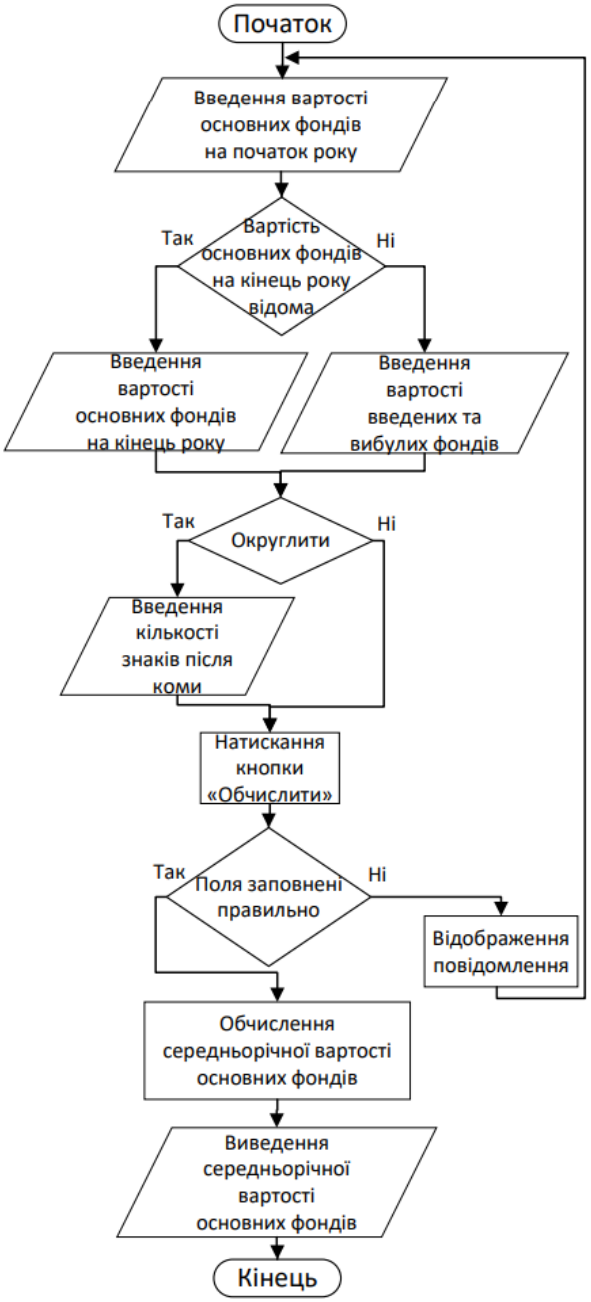

Рис. 2.5. Блок-схема алгоритму роботи екрану розрахунку середньорічної вартості основних фондів

Блок-схема алгоритму роботи екрану, призначеного для обчислення коефіцієнта вибуття, представлена на рис. 2.6.

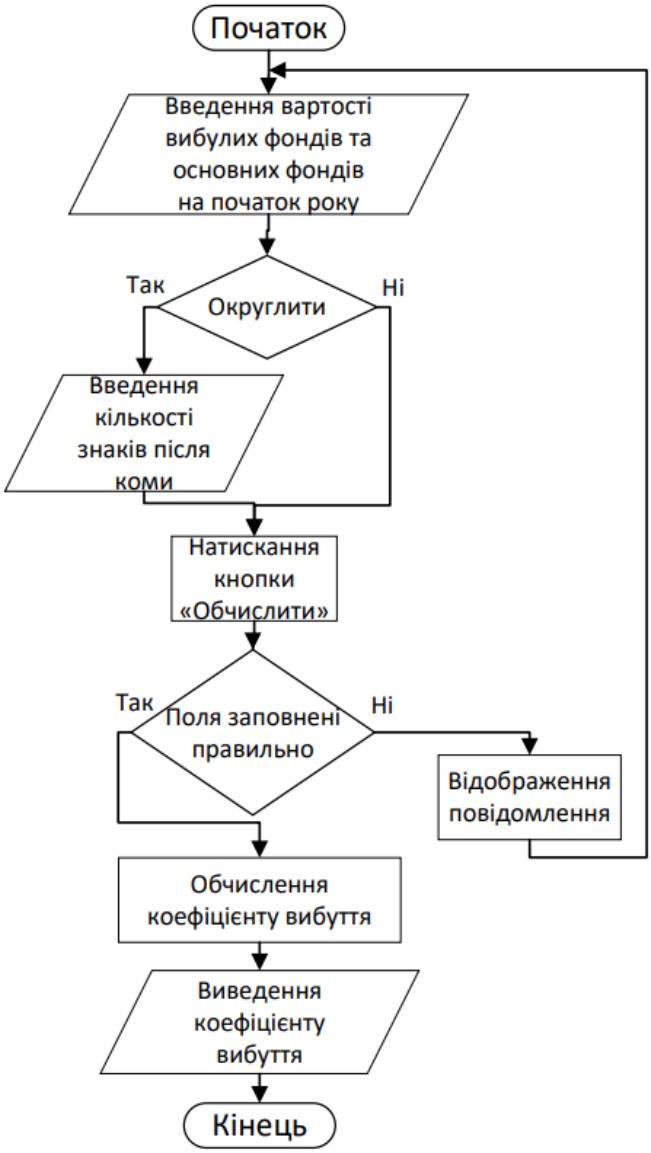

Рис. 2.6. Блок-схема алгоритму роботи екрану розрахунку коефіцієнта вибуття

Блок-схема алгоритму роботи екрану, призначеного для обчислення коефіцієнта відновлення, представлена на рис. 2.7.

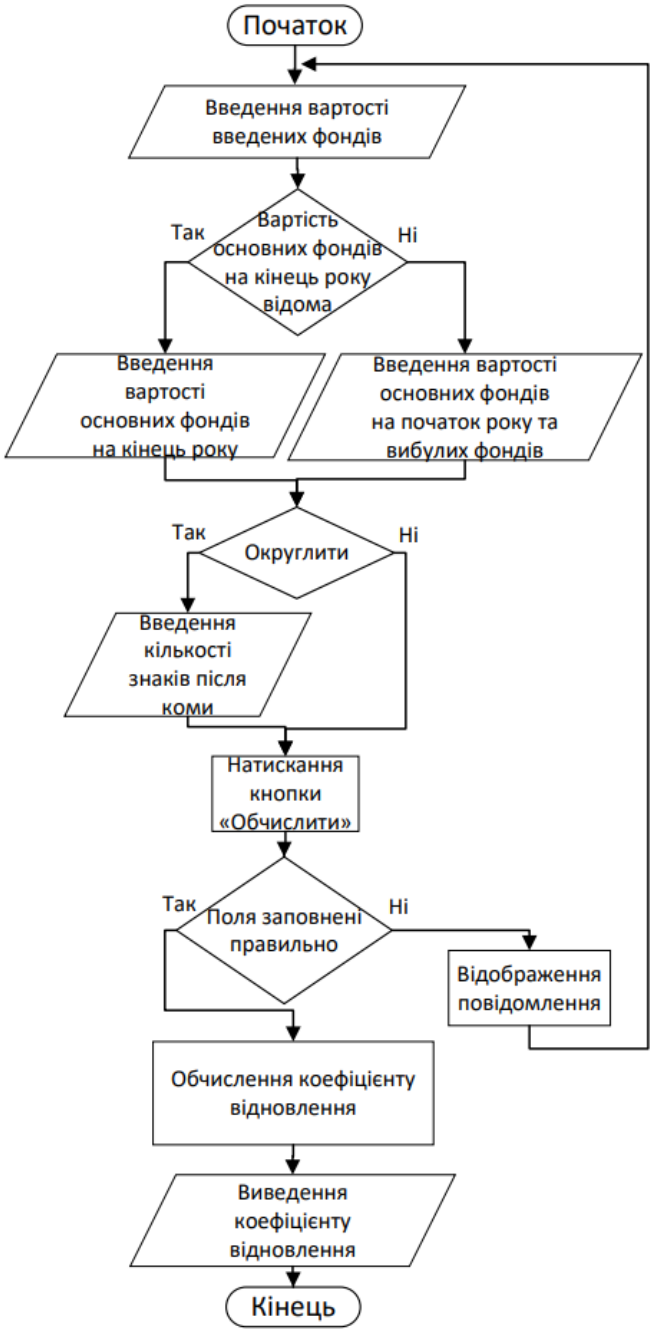

Рис. 2.7. Блок-схема алгоритму роботи екрану розрахунку коефіцієнта відновлення

Блок-схема алгоритму роботи екрану, призначеного для розрахунку коефіцієнта зносу, представлена на рис. 2.8.

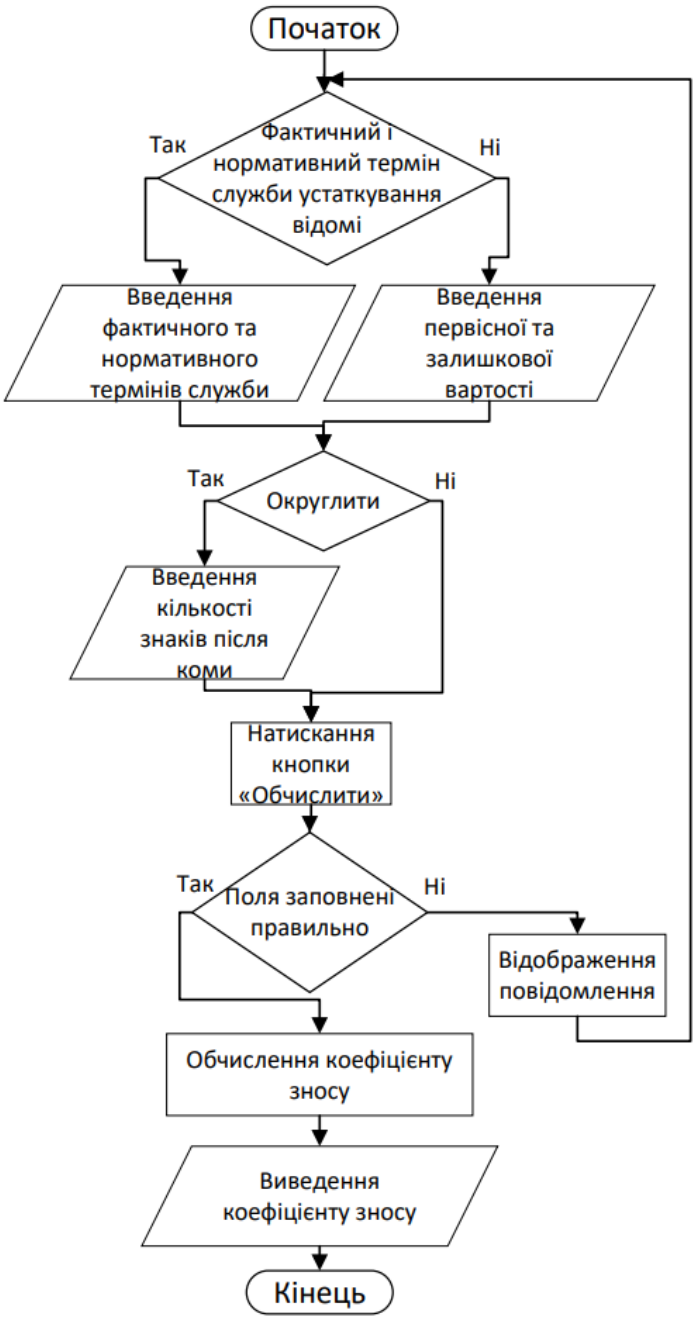

Рис. 2.8. Блок-схема алгоритму роботи екрану розрахунку коефіцієнта зносу

Блок-схема алгоритму роботи екрану, призначеного для обчислення номінального (режимного) фонду, представлена на рис. 2.9.

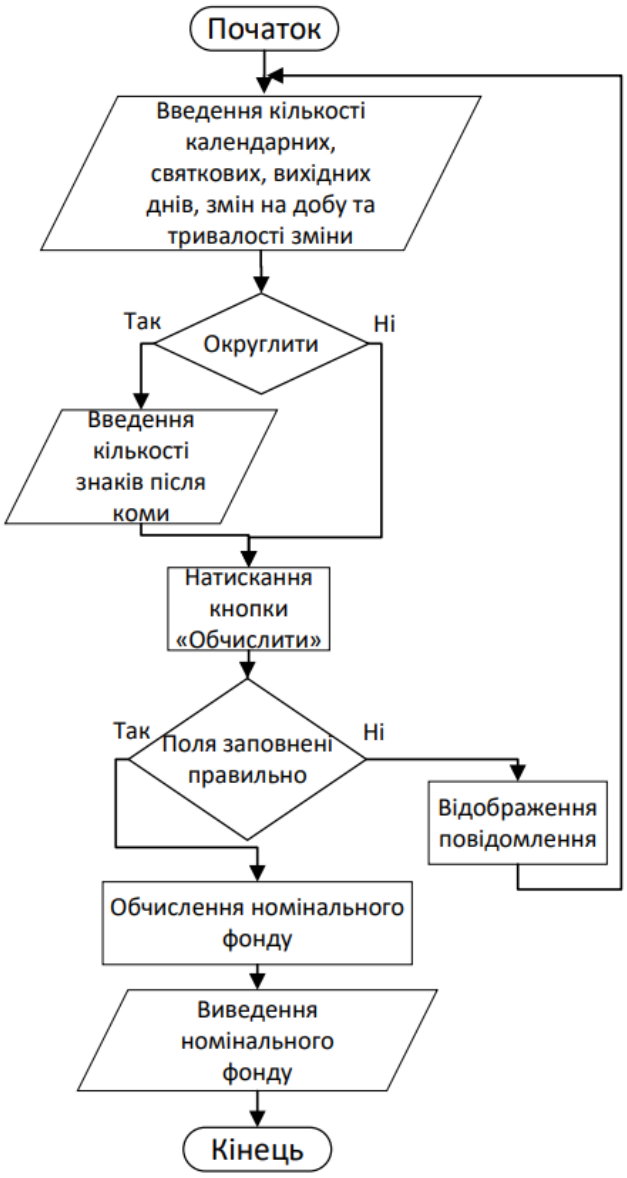

Рис. 2.9. Блок-схема алгоритму роботи екрану розрахунку номінального (режимного) фонду

Блок-схема алгоритму роботи екрану, призначеного для обчислення дійсного (ефективного) фонду, представлена на рис. 2.10.

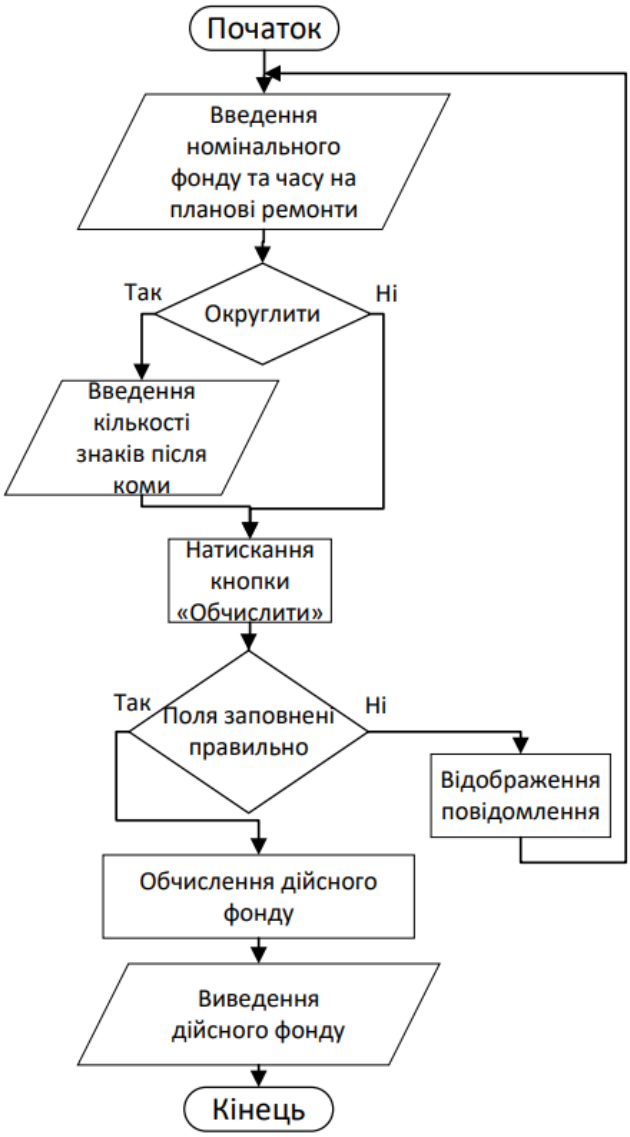

Рис. 2.10. Блок-схема алгоритму роботи екрану розрахунку дійсного (ефективного) фонду

Блок-схема алгоритму роботи екрану, призначеного для обчислення коефіцієнта екстенсивного використання устаткування, представлена на рис. 2.11.

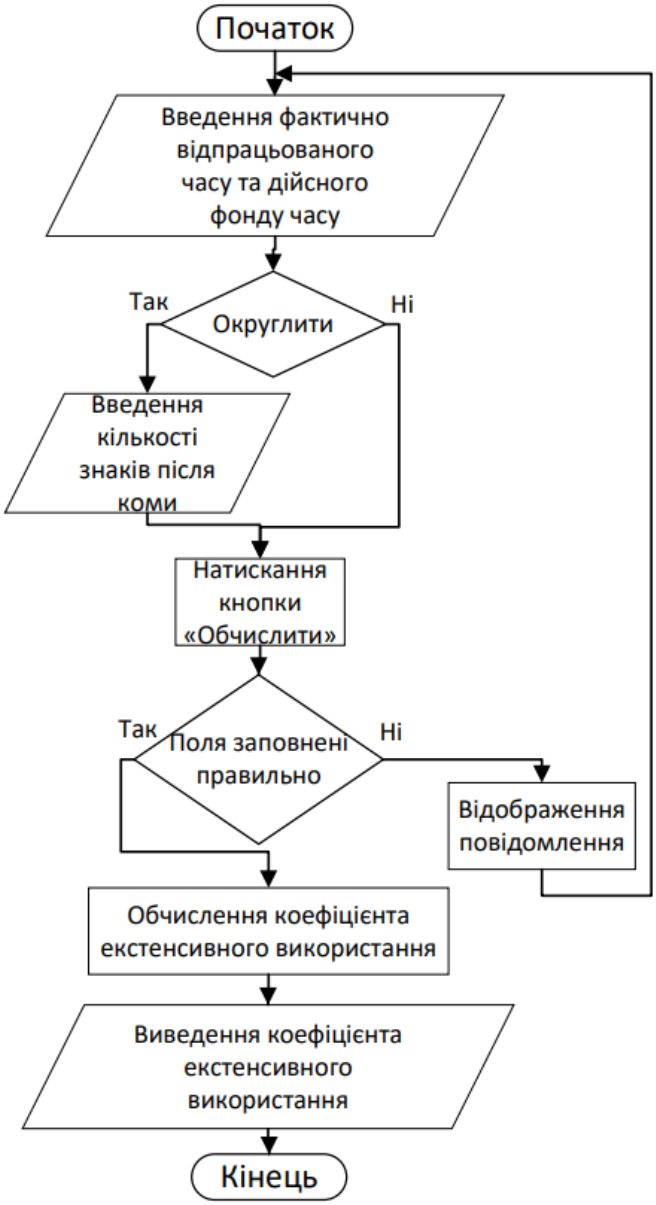

Рис. 2.11. Блок-схема алгоритму роботи екрану розрахунку коефіцієнта екстенсивного використання устаткування

Блок-схема алгоритму роботи екрану, призначеного для обчислення коефіцієнта інтенсивного використання устаткування, представлена на рис. 2.12.

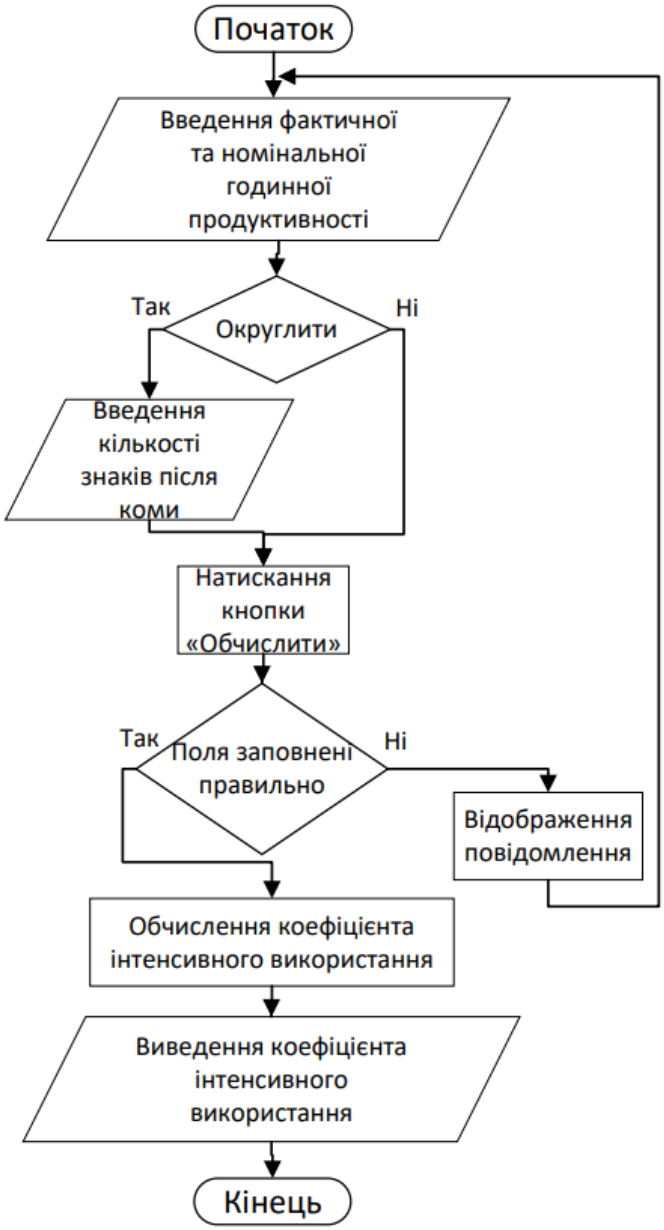

Рис. 2.12. Блок-схема алгоритму роботи екрану розрахунку коефіцієнта інтенсивного використання устаткування

Блок-схема алгоритму роботи екрану, призначеного для розрахунку фондовіддачі, представлена на рис. 2.13.

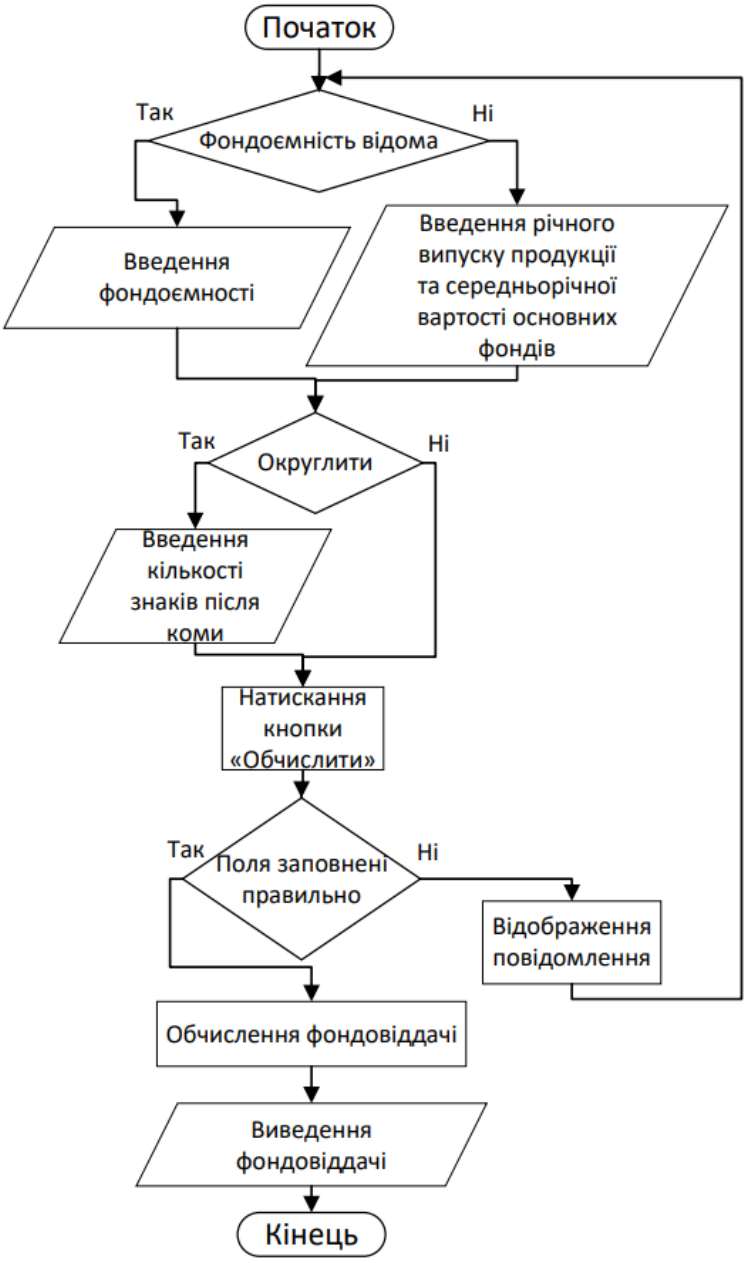

Рис. 2.13. Блок-схема алгоритму роботи екрану розрахунку фондовіддачі

Блок-схема алгоритму роботи екрану, призначеного для обчислення фондоємності, представлена на рис. 2.14.

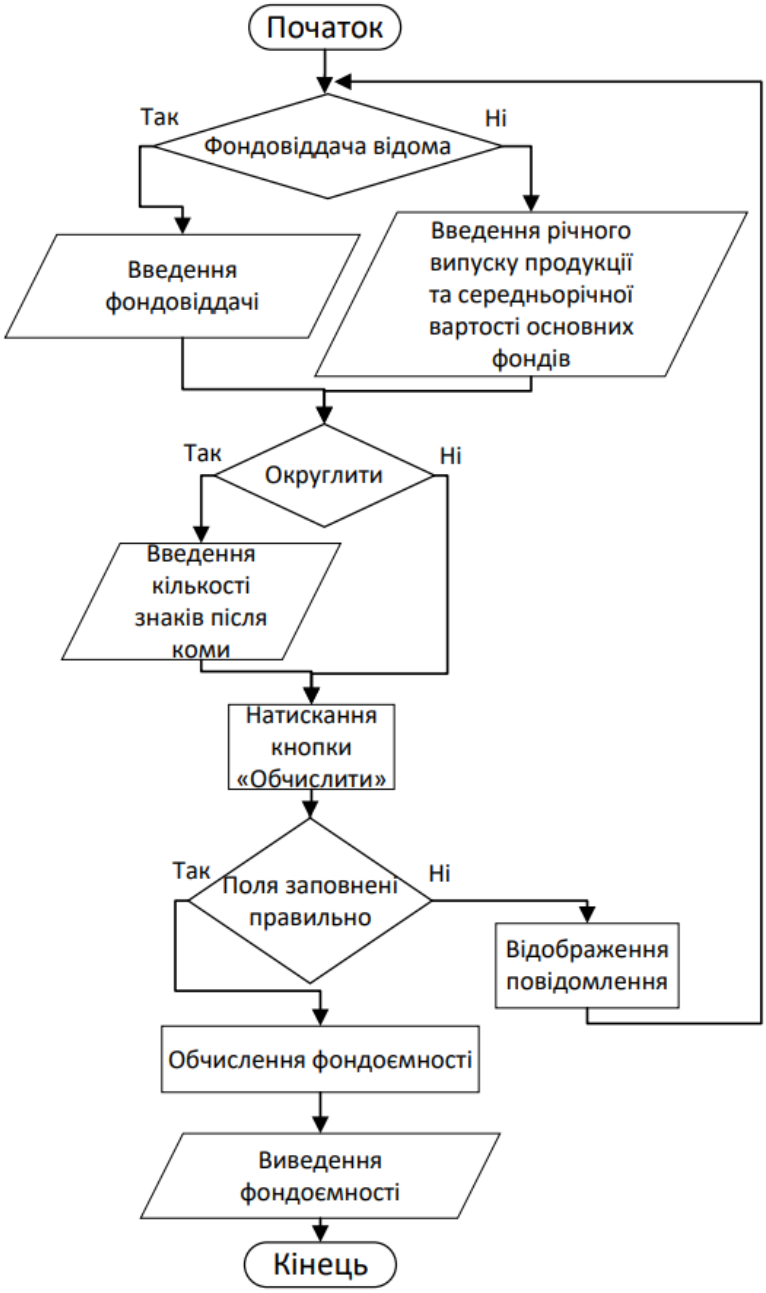

Рис. 2.14. Блок-схема алгоритму роботи екрану розрахунку фондоємності

Блок-схема алгоритму роботи екрану, призначеного для обчислення показника фондоозброєності, представлена на рис. 2.15.

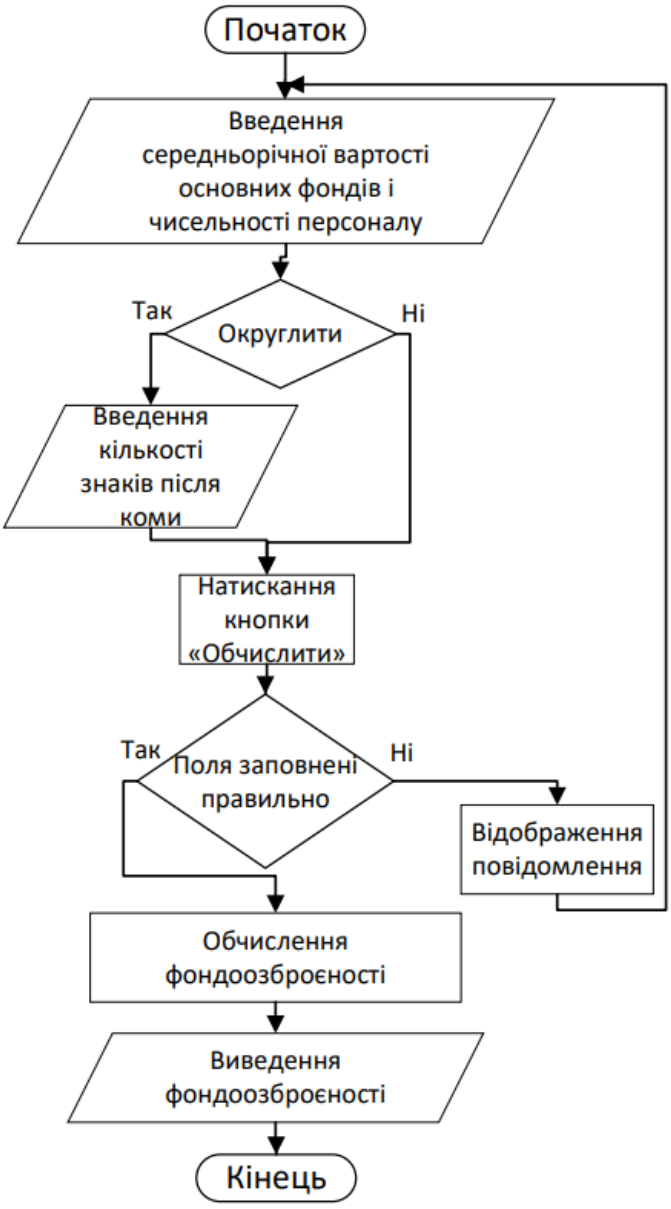

Рис. 2.15. Блок-схема алгоритму роботи екрану розрахунку фондоозброєності

Блок-схема алгоритму роботи екрану, призначеного для обчислення інтегрального коефіцієнта використання устаткування, представлена на рис. 2.16.

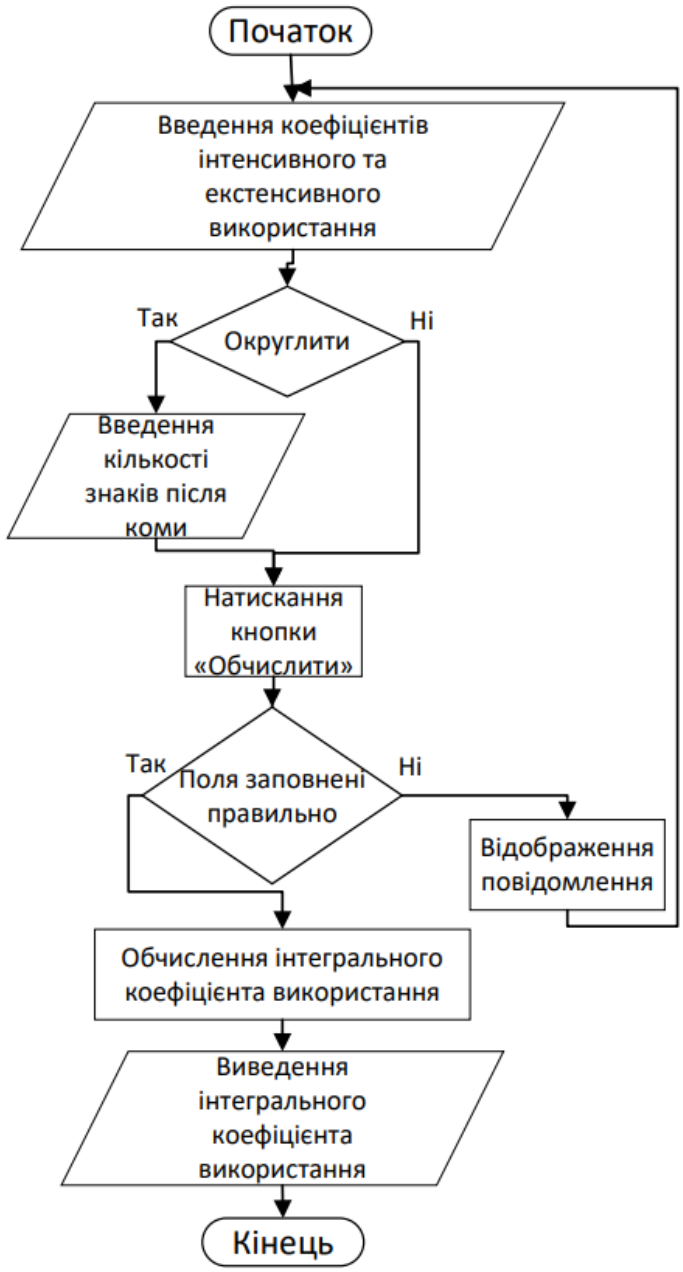

Рис. 2.16. Блок-схема алгоритму роботи екрану розрахунку інтегрального коефіцієнта використання устаткування

Блок-схема алгоритму роботи екрану, призначеного для розрахунку показника рентабельності основних фондів, представлена на рис. 2.17.

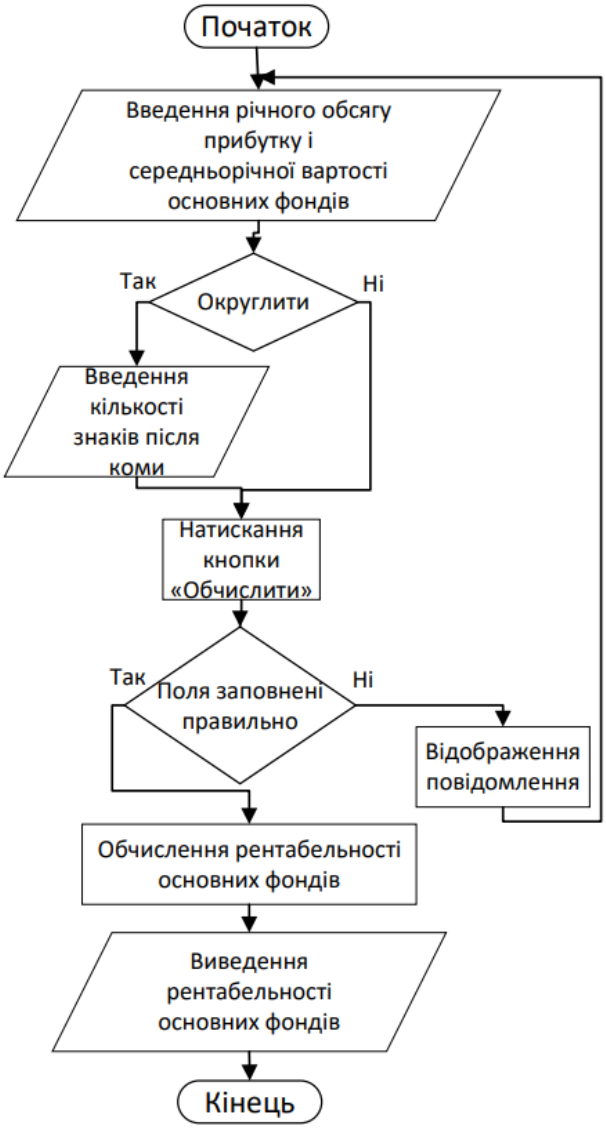

Рис. 2.17. Блок-схема алгоритму роботи екрану розрахунку рентабельності основних фондів

Оскільки після коректного введення вхідних даних користувачем відразу виконуються обчислення, а також виводиться результат цих обчислень, то база даних для функціонування даного ПЗ не використовується.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Введення вхідних даних здійснюється користувачем у текстові поля, що призначені для цих даних. Дані, що введені користувачем, перевіряються на відповідність вимогам до цих даних. Якщо значення, введене користувачем, не є

числом або не належить до діапазону можливих допустимих значень, то користувачу виводиться відповідне повідомлення.

Вихідними даними, що надаються мобільним додатком, є результати здійснених розрахунків відповідно до обраного користувачем методу розрахунку амортизації (якщо обчислюється амортизація) або економічного показника.

На екрані для розрахунку амортизації прямолінійним методом вводяться такі вхідні дані:

– первісна вартість (якщо вона відома, інакше – витрати на придбання устаткування, транспортування, установку та на будівельні роботи для споруд);
– ліквідаційна вартість;
– термін корисного використання;
– кількість знаків після коми (якщо опція округлення обрана).

Вихідними даними для цього екрану є такі:

– річна сума амортизації;
– місячна сума амортизації.

На екрані для обчислення амортизації методом зменшення залишкової вартості вводяться наступні вхідні дані:

– первісна вартість (якщо відома, в іншому випадку – витрати на придбання устаткування, транспортування, установку, а також на будівельні роботи для споруд);
– ліквідаційна вартість;
– термін корисного використання;
– річна норма амортизації (якщо вона відома);
– кількість знаків після коми (якщо обрано опцію округлення).

Вихідні дані для цього екрану (окремо для кожного року експлуатації):

– річна сума амортизаційних відрахувань;
– місячна сума амортизаційних відрахувань;
– накопичена амортизація;
– залишкова вартість.

На екранах для розрахунку амортизації методом прискореного зменшення залишкової вартості, а також кумулятивним методом, вводяться такі ж вхідні дані, як і на екрані для здійснення обчислень амортизації прямолінійним методом. Вихідними даними для цих екранів є ті ж дані, що й для екрану здійснення розрахунку амортизації за методом зменшення залишкової вартості.

На екрані для обчислення середньорічної вартості основних фондів вводяться вхідні дані:

– вартість основних фондів на початок року;

– вартість основних фондів на кінець року (якщо відома, інакше – окремо вартість введених та вибулих фондів);

– кількість знаків після коми (якщо опція округлення обрана).

Вихідними даними для цього екрану є середньорічна вартість основних фондів.

На екрані для розрахунку коефіцієнта вибуття вводяться такі вхідні дані:

– вартість вибулих фондів;

– вартість основних фондів на початок року;

– кількість знаків після коми (якщо опція округлення обрана).

Вихідними даними для цього екрану є коефіцієнт вибуття.

На екрані для обчислення коефіцієнта відновлення вводяться вхідні дані:

– вартість введених фондів;

– вартість основних фондів на кінець року (якщо відома, інакше – вартість основних фондів на початок року та вартість вибулих фондів);

– кількість знаків після коми (якщо опція округлення обрана).

Вихідними даними для цього екрану є коефіцієнт відновлення.

Для екрану з обчисленням коефіцієнта зносу вводяться такі вхідні дані:

– фактичний та нормативний терміни служби устаткування (якщо вони відомі, в іншому випадку – первісна та залишкова вартість);

– кількість знаків після коми (якщо обрана опція для округлення).

Вихідні дані для цього екрану - це коефіцієнт зносу.

На екрані для розрахунку номінального фонду необхідно ввести наступні

вхідні дані:

– календарні дні;

– святкові дні;

– вихідні дні;

– кількість змін на добу;

– тривалість зміни у годинах;

– кількість знаків після коми (якщо опція округлення обрана).

Вихідними даними для цього екрану є показник номінального фонду.

На екрані для розрахунку дійсного фонду вводяться такі вхідні дані:

– номінальний фонд;

– відсоток часу на планові ремонти;

– кількість знаків після коми (якщо обрано опцію округлення).

Вихідними даними для цього екрану є показник дійсного фонду.

Для екрану з обчисленням коефіцієнта екстенсивного використання необхідно ввести вхідні дані:

– фактично відпрацьований час;

– дійсний фонд часу в годинах;

– кількість знаків після коми (якщо обрана опція для округлення).

Вихідні дані для цього екрану - коефіцієнт екстенсивного використання.

Для екрану з розрахунком коефіцієнта інтенсивного використання необхідно ввести наступні вхідні дані:

– фактична годинна продуктивність;

– номінальна годинна продуктивність;

– кількість знаків після коми (якщо опція для округлення обрана).

Вихідні дані для цього екрану - це коефіцієнт інтенсивного використання.

На екрані для обчислення показника фондовіддачі вводяться вхідні дані:

– фондоємність (якщо вона відома, інакше – обсяг річного випуску продукції та середньорічна вартість основних фондів);

– кількість знаків після коми (якщо обрано опцію округлення).

Вихідними даними для цього екрану є показник фондовіддачі.

На екрані для обчислення показника фондоємності вводяться наступні вхідні дані:

– фондовіддача (якщо вона відома, інакше – обсяг річного випуску продукції, а також середньорічна вартість основних фондів);

– кількість знаків після коми (якщо обрано опцію для округлення).

Вихідними даними для цього екрану є показник фондоємності.

На екрані для розрахунку показника фондоозброєності необхідно ввести такі вхідні дані:

– середньорічна вартість основних фондів;

– чисельність персоналу;

– кількість знаків після коми (якщо обрано опцію округлення).

Вихідними даними для цього екрану є показник фондоозброєності.

Для екрану з обчисленням інтегрального коефіцієнта використання устаткування вводяться такі вхідні дані:

– коефіцієнт інтенсивного використання;

– коефіцієнт екстенсивного використання;

– кількість знаків після коми (якщо опція для округлення обрана).

Вихідні дані для цього екрану - інтегральний коефіцієнт використання устаткування.

На екрані для обчислення показника рентабельності основних фондів вводяться вхідні дані:

– річний обсяг прибутку;

– середньорічна вартість основних фондів;

– кількість знаків після коми (якщо обрано опцію округлення).

Вихідні дані цього екрану – це показник рентабельності основних фондів.

Отже, для виконання розрахунку та аналізу економічних показників діяльності підприємства використовуються основні економіко-математичні моделі (2.1 – 2.7).

2.6. Опис розробленої системи

В даному підрозділі наведено опис технічних та програмних засобів, що були використані для розробки мобільного додатку, описується процес виклику даного ПЗ, а також представляється графічний інтерфейс застосунку.

2.6.1. Використані технічні засоби

Для створення даного ПЗ було використано такі технічні засоби:

- процесор Intel Core i3-8145U;
- використане місце на диску – 2,5 ГБ;
- використана оперативна пам’ять – 2 ГБ;
- клавіатура;
- маніпулятор «миша»;
- ОС Microsoft Windows 10.

2.6.2. Використані програмні засоби

До програмних засобів, які використовувались під час розробки додатку, належать наступні:

- середовище програмування PyCharm Community Edition 2022.3.2;
- мова програмування Python;
- фреймворк Kivy;
- набір офісних програм Microsoft Office.

2.6.3. Виклик та завантаження програми

Для запуску даного ПЗ на пристроях з різними операційними системами використовуються файли з різним розширенням. Таким чином, якщо користувач запускає додаток на девайсі з ОС Android, то необхідно використовувати APK-

файл, а якщо, наприклад, на Windows, то ПЗ необхідно запускати через файл з розширенням .exe.

2.6.4. Опис інтерфейсу користувача

Після запуску додатка відкривається головний екран, який містить список можливих опцій, які можна обрати для виконання розрахунків амортизації та економічних показників. Графічний інтерфейс даного екрану зображено на рис. 2.18.

Рис. 2.18. Головний екран додатку

Зовнішній вигляд екрану для розрахунку амортизації прямолінійним методом подано на рис. 2.19.

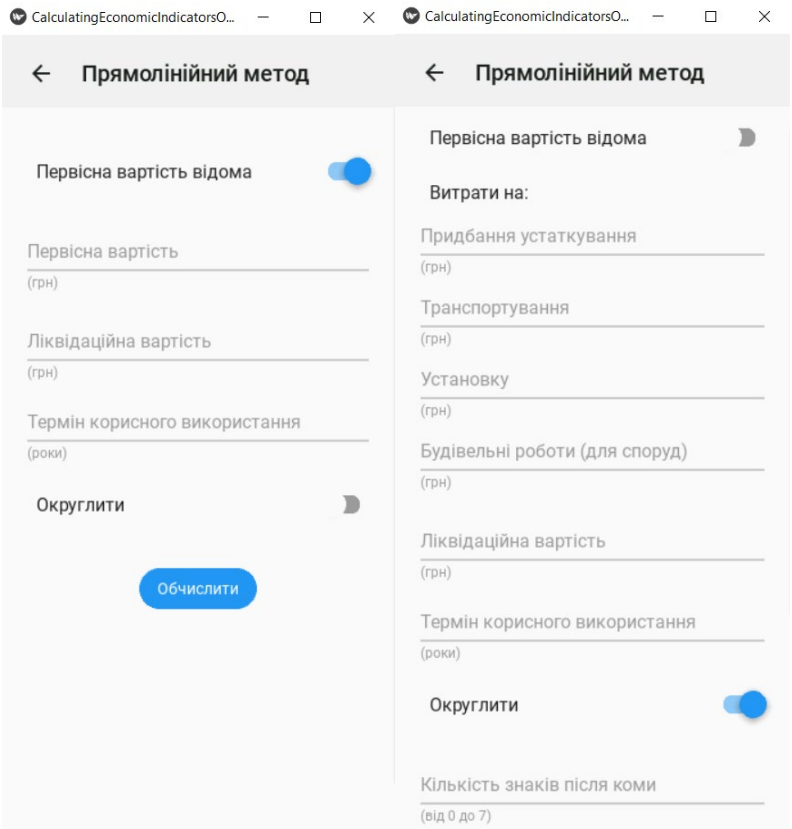

Рис. 2.19. Екран для розрахунку амортизації прямолінійним методом

Варто зазначити, що на цьому екрані та на інших екранах для виконання обчислень наявні перемикачі, які дозволяють обрати різні варіанти вхідних даних, округлити результат тощо. Таким чином, в залежності від того, які перемикачі є активними, а які є вимкненими, відображаються відповідні текстові поля для введення даних.

Отже, на даному екрані первісна вартість вводиться користувачем, якщо вона відома та відповідний перемикач є активним, в інакшому випадку – користувач вводить вхідні дані для розрахунку первісної вартості.

Приклад здійснення обчислень амортизації прямолінійним методом зображений на рис. 2.20.

Крім того, якщо користувач вводить дані, які не належать до діапазону можливих значень для даного текстового поля, то відображається діалогове вікно з відповідним текстом. Можливі варіанти діалогового вікна, що відображається при введенні некоректних даних, показані на рис. 2.21.

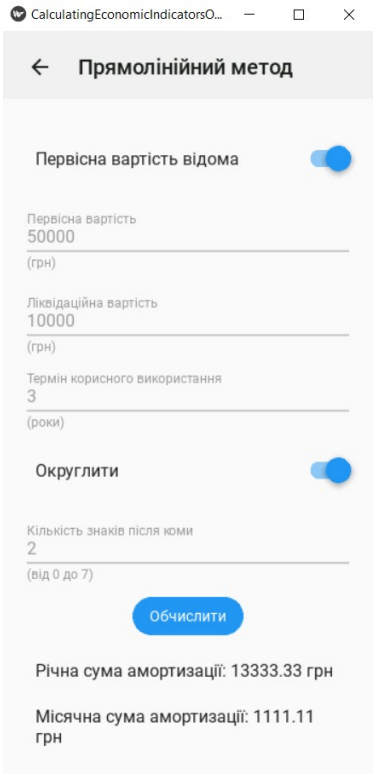

Рис. 2.20. Приклад обчислення амортизації прямолінійним методом

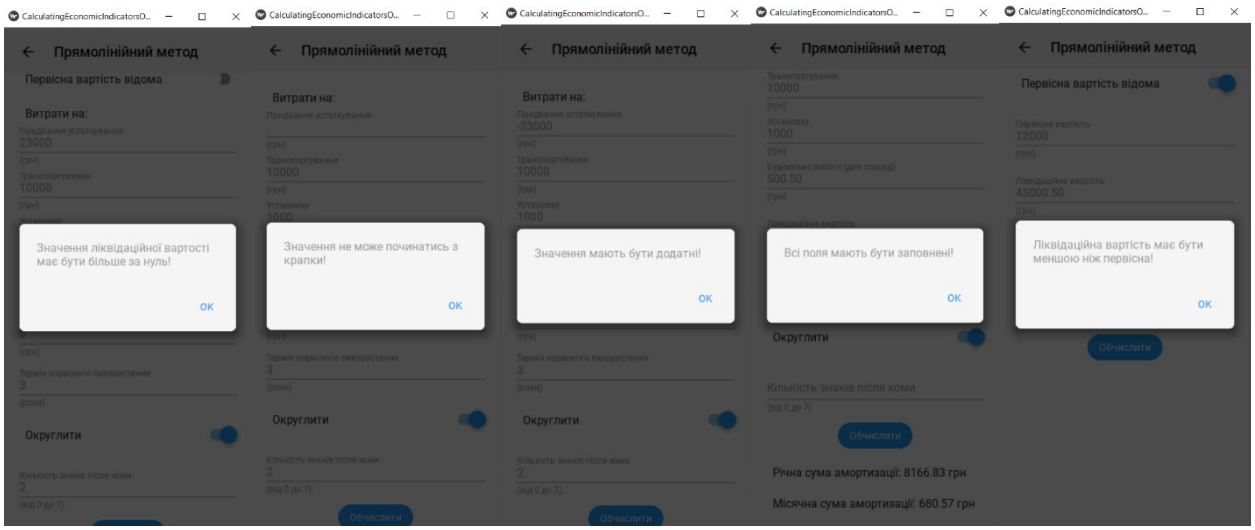

Рис. 2.21. Діалогові вікна при введенні некоректних даних

Екран для обчислення амортизації методом зменшення залишкової вартості зображено на рис. 2.22.

Приклад розрахунку амортизації методом зменшення залишкової вартості зображений на рис. 2.23.

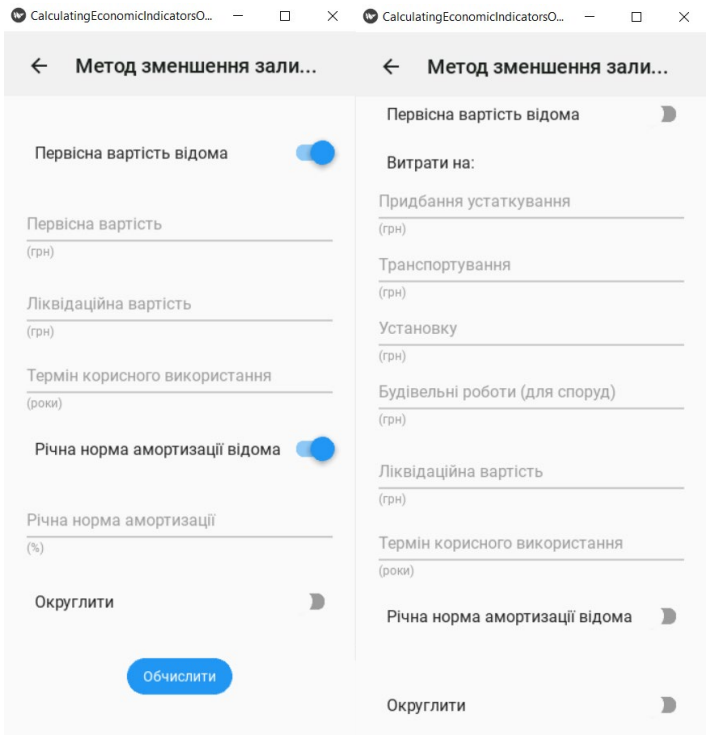

Рис. 2.22. Екран для розрахунку амортизації методом зменшення залишкової вартості

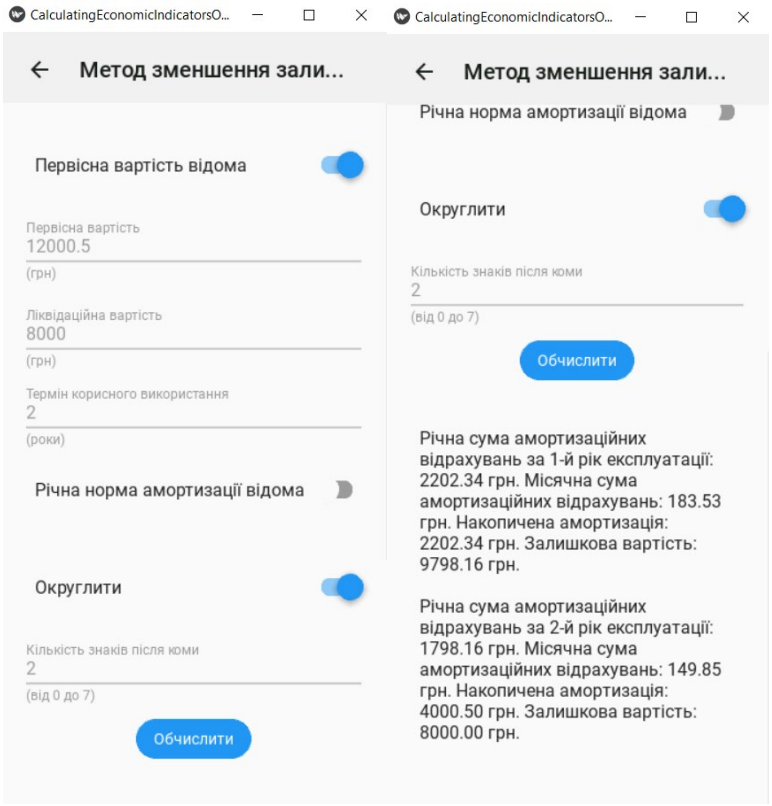

Рис. 2.23. Приклад обчислення амортизації методом зменшення залишкової вартості

Екран для розрахунку амортизації методом прискореного зменшення залишкової вартості наведено на рис. 2.24.

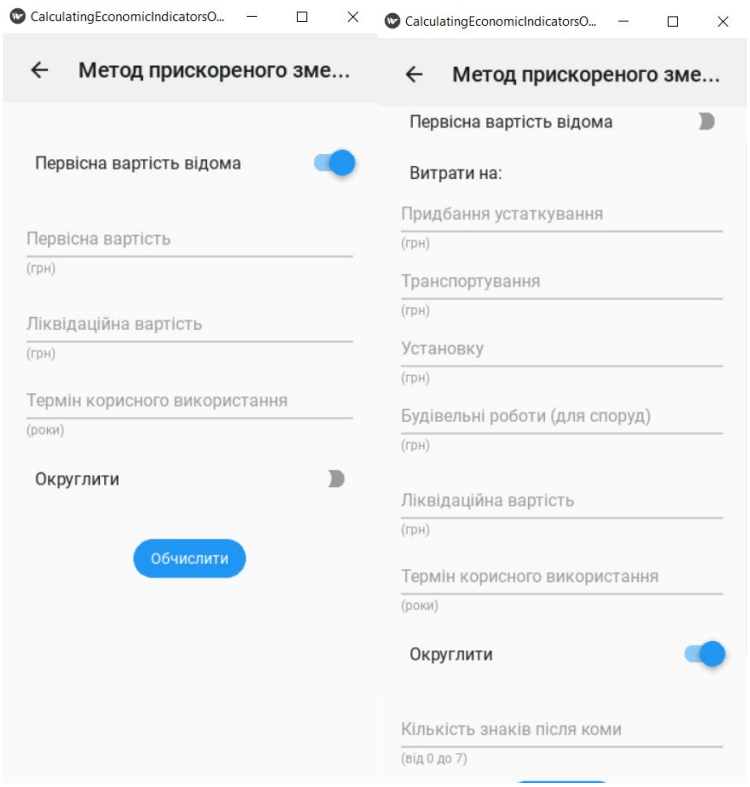

Рис. 2.24. Екран для розрахунку амортизації методом прискореного зменшення залишкової вартості

Приклад розрахунку амортизації методом прискореного зменшення залишкової вартості показано на рис. 2.25.

Екран для обчислення амортизації кумулятивним методом зображено на рис. 2.26. Приклад розрахунку амортизації кумулятивним методом зображений на рис. 2.27.

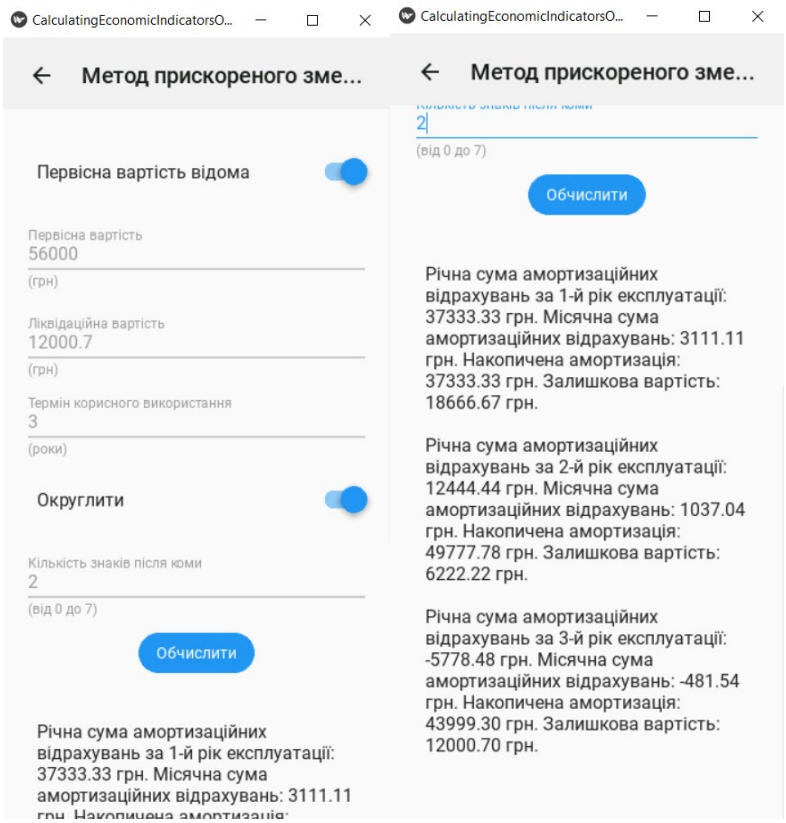

Рис. 2.25. Приклад обчислення амортизації методом прискореного зменшення залишкової вартості

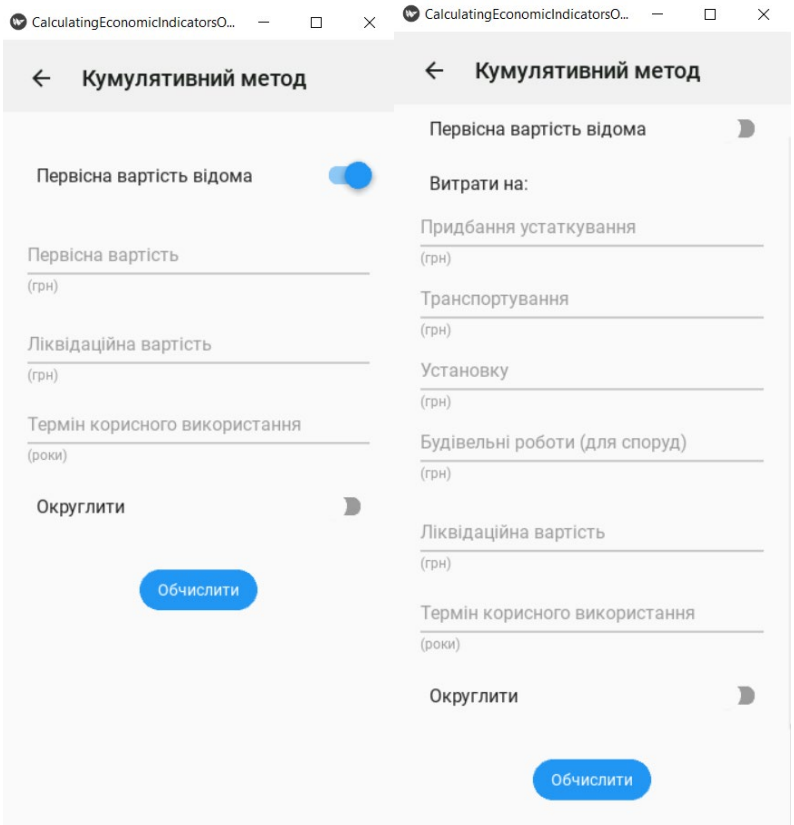

Рис. 2.26. Екран для розрахунку амортизації кумулятивним методом

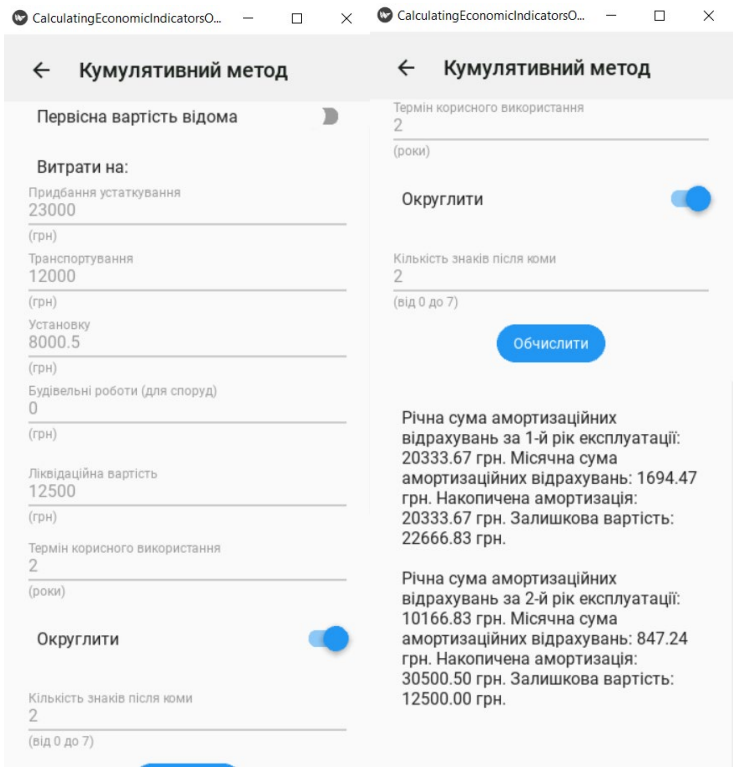

Рис. 2.27. Приклад обчислення амортизації кумулятивним методом

Екран для розрахунку середньорічної вартості основних фондів наведено на рис. 2.28.

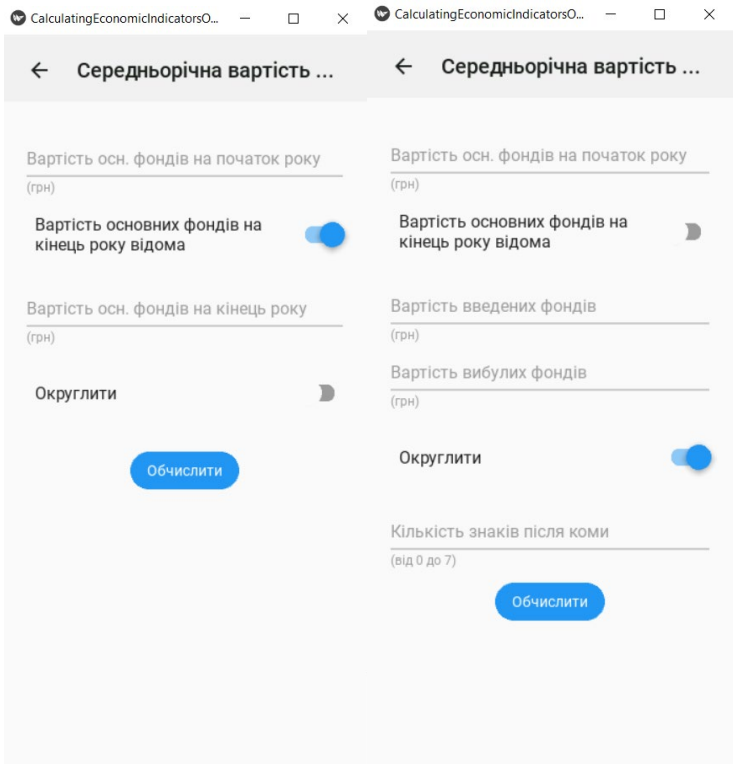

Рис. 2.28. Екран для розрахунку середньорічної вартості основних фондів

Приклад розрахунку середньорічної вартості основних фондів показано на рис. 2.29. Екран для розрахунку коефіцієнта вибуття, а також приклад виконання обчислень, зображено на рис. 2.30.

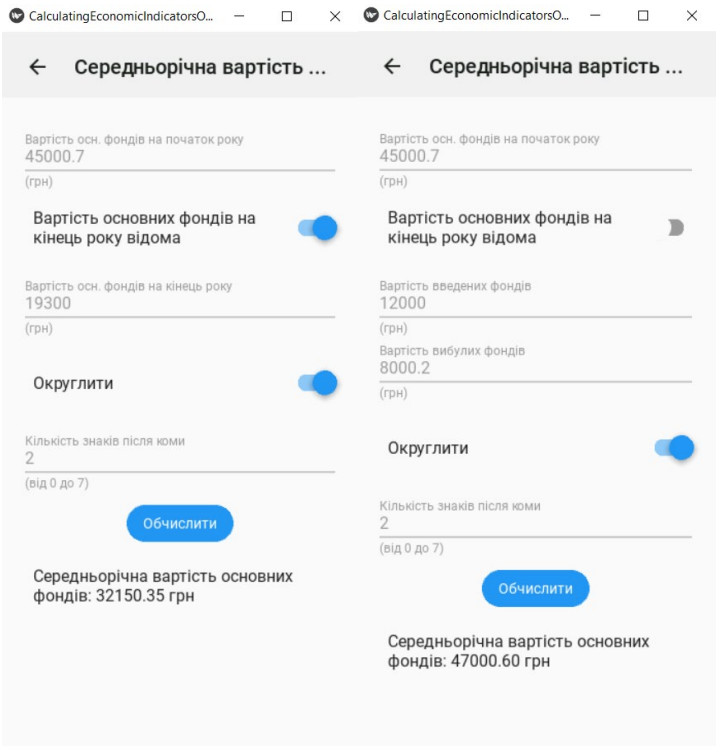

Рис. 2.29. Приклад обчислення середньорічної вартості основних фондів

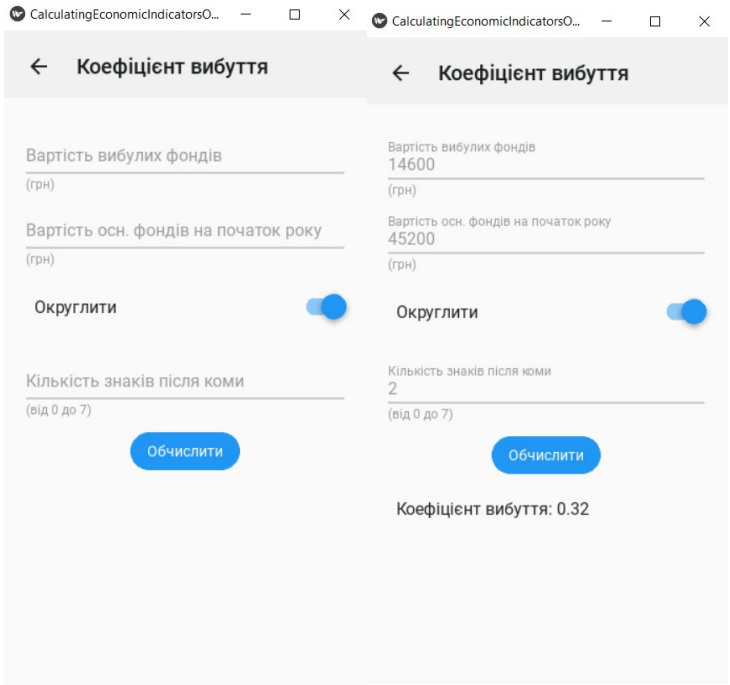

Рис. 2.30. Екран для розрахунку коефіцієнта вибуття та приклад обчислень

Екран для обчислення коефіцієнта відновлення наведено на рис. 2.31. Приклад розрахунку коефіцієнта відновлення показано на рис. 2.32.

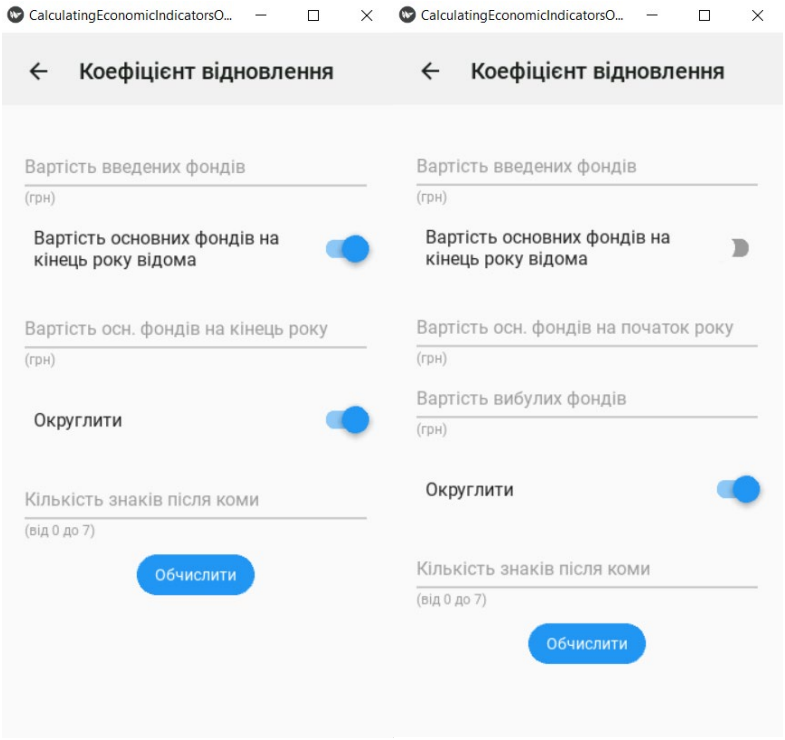

Рис. 2.31. Екран для обчислення коефіцієнта відновлення

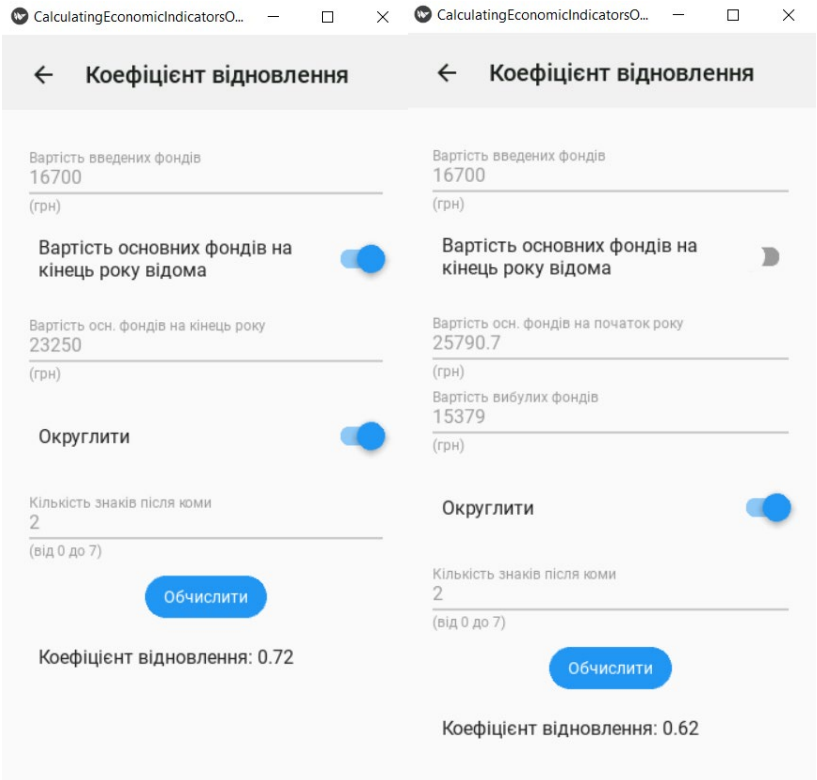

Рис. 2.32. Приклад обчислення коефіцієнта відновлення

Екран для обчислення коефіцієнта зносу наведено на рис. 2.33. Приклад розрахунку коефіцієнта зносу показано на рис. 2.34.

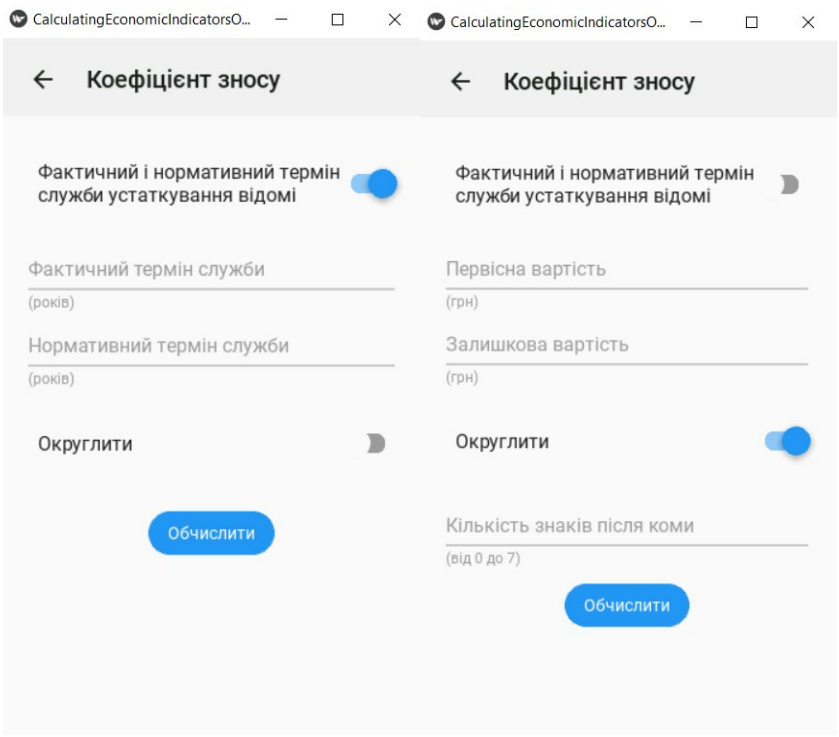

Рис. 2.33. Екран для обчислення коефіцієнта зносу

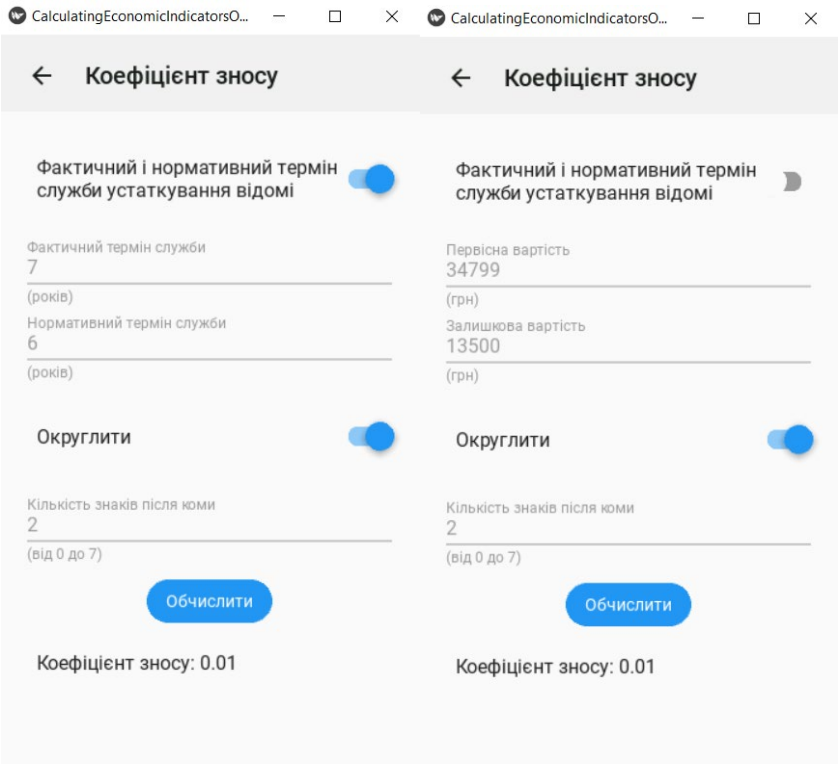

Рис. 2.34. Приклад обчислення коефіцієнта зносу

Екран для обчислення номінального фонду, а також приклад виконання розрахунку, зображено на рис. 2.35. Екран для розрахунку дійсного фонду, а також приклад виконання обчислень, зображено на рис. 2.36.

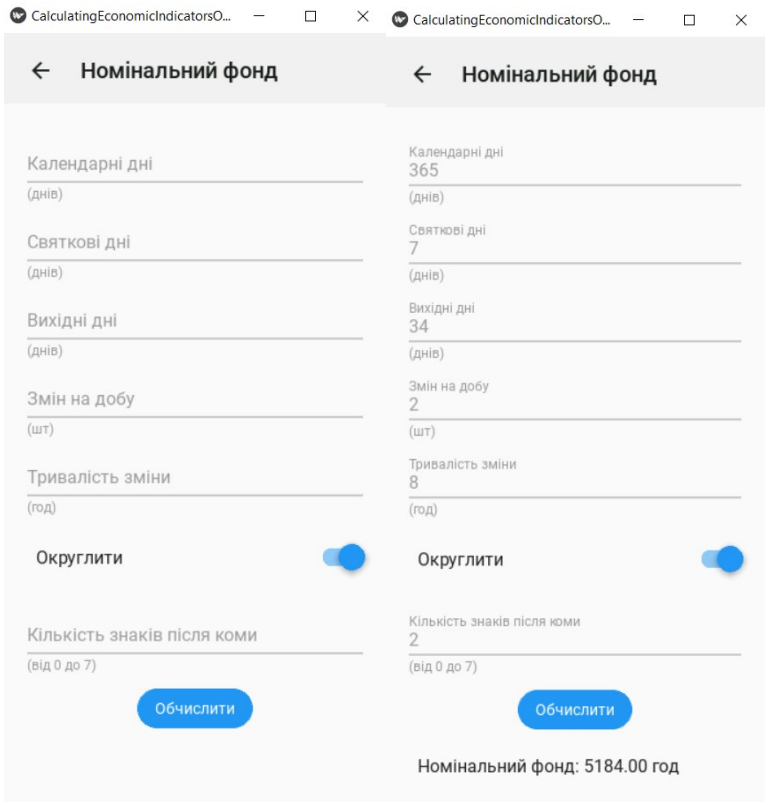

Рис. 2.35. Екран для розрахунку номінального фонду

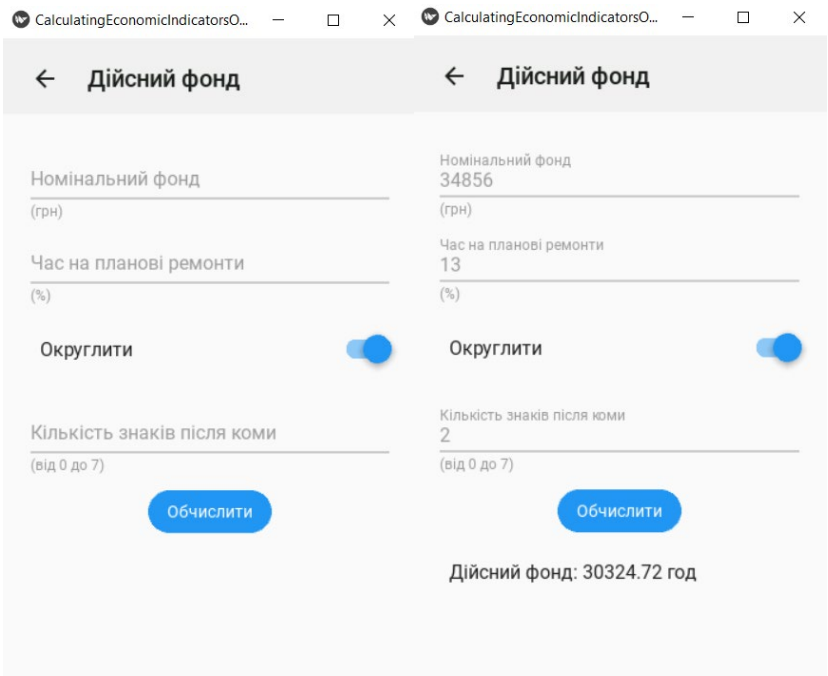

Рис. 2.36. Екран для розрахунку дійсного фонду

Екран для обчислення коефіцієнта екстенсивного використання, а також приклад виконання розрахунку, зображено на рис. 2.37. Екран для розрахунку коефіцієнта інтенсивного використання, а також приклад виконання обчислень, зображено на рис. 2.38.

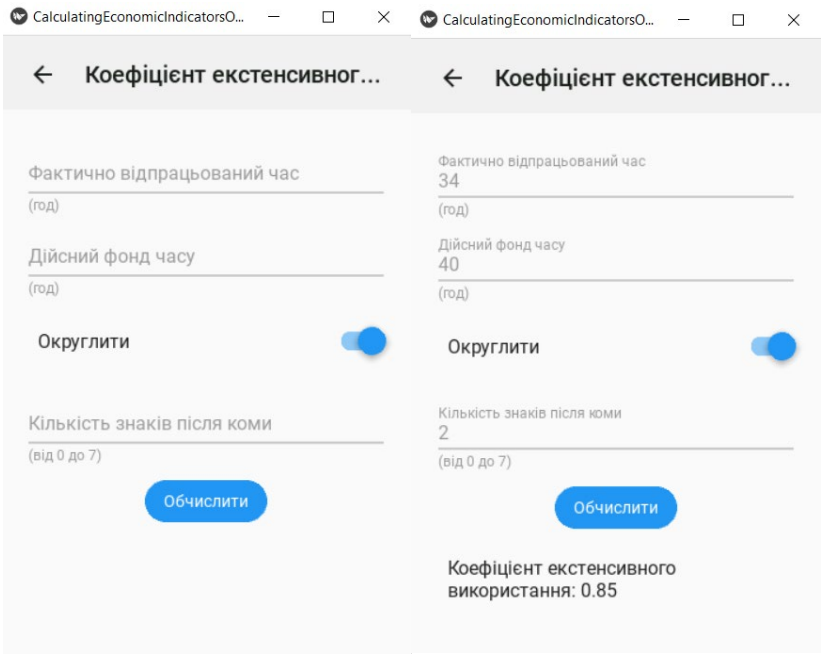

Рис. 2.37. Екран для розрахунку коефіцієнта екстенсивного використання

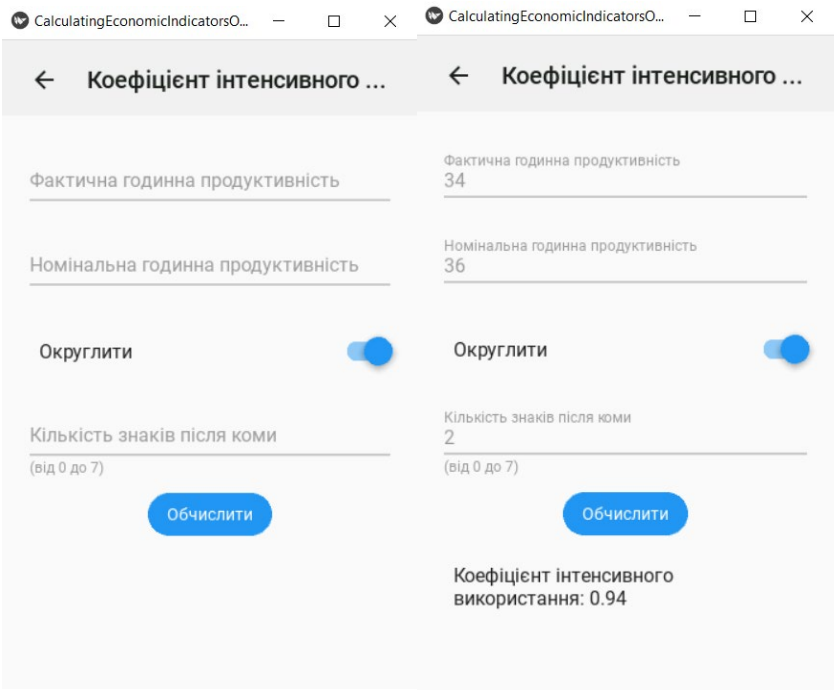

Рис. 2.38. Екран для розрахунку коефіцієнта інтенсивного використання

Екран для обчислення фондовіддачі наведено на рис. 2.39. Приклад розрахунку фондовіддачі показано на рис. 2.40.

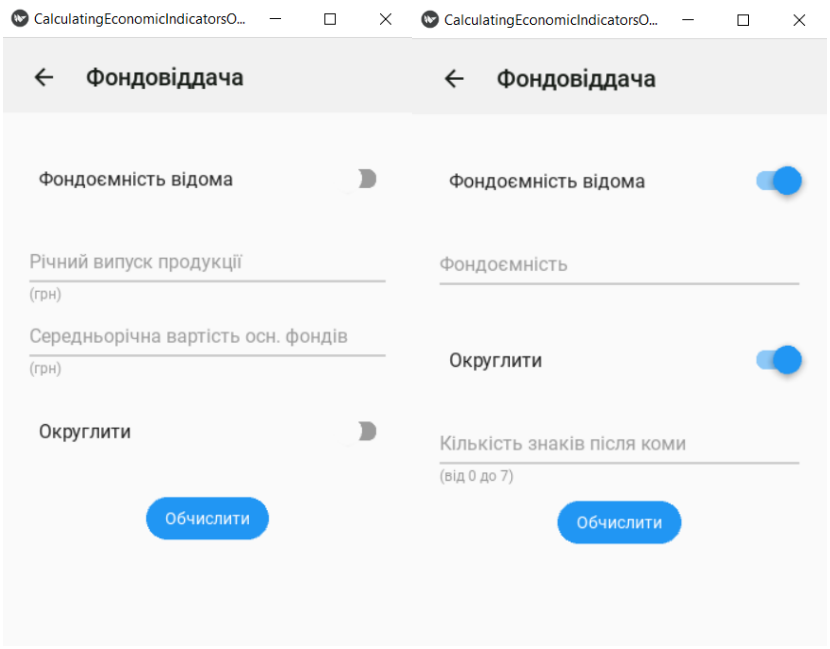

Рис. 2.39. Екран для обчислення фондовіддачі

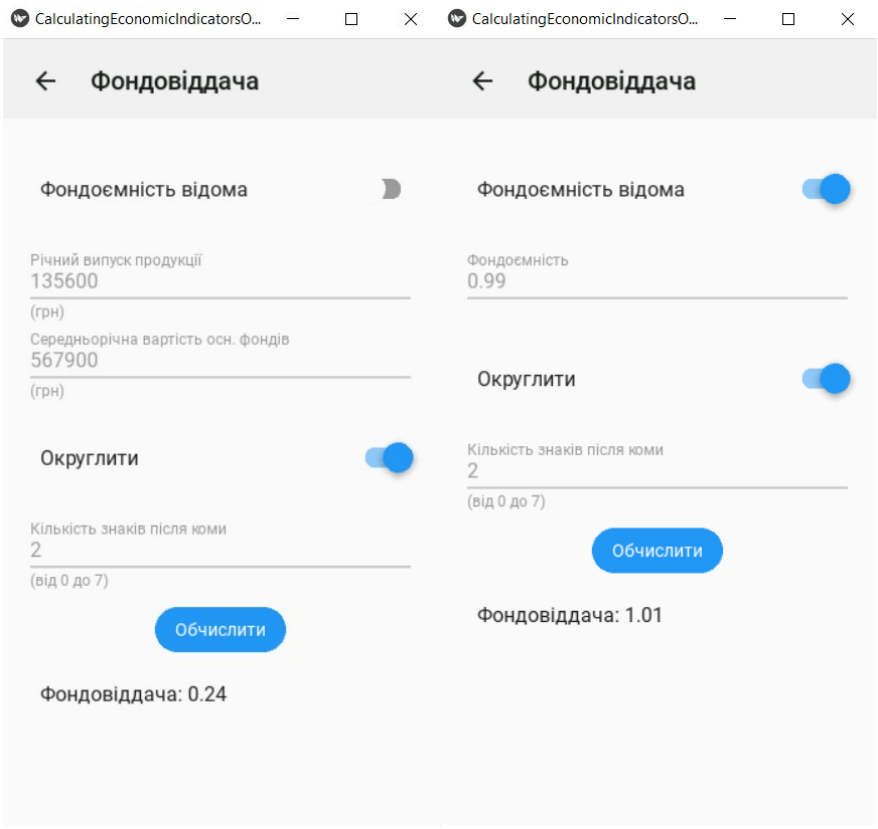

Рис. 2.40. Приклад обчислення фондовіддачі

Екран для обчислення фондоємності наведено на рис. 2.41. Приклад розрахунку фондоємності показано на рис. 2.42.

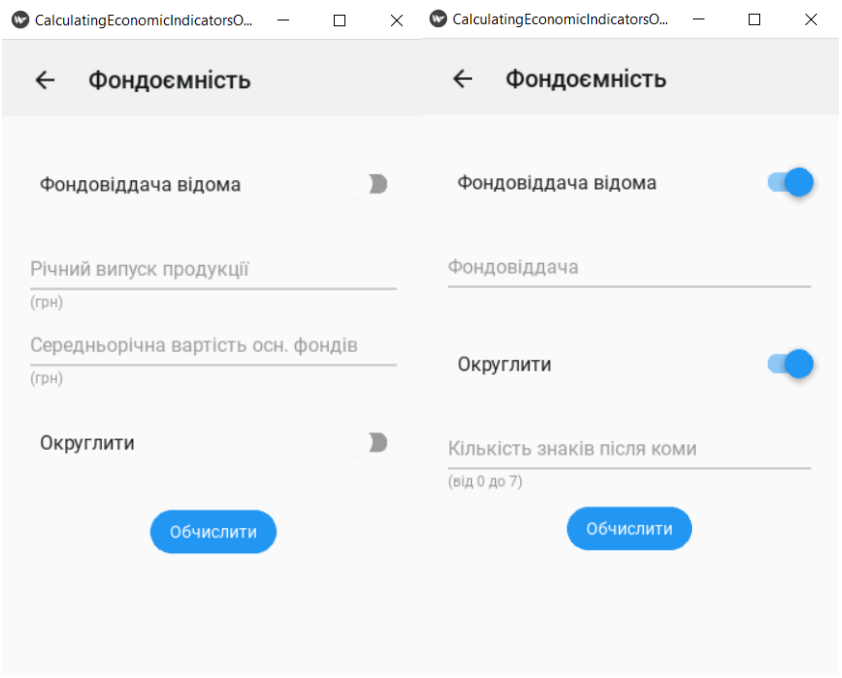

Рис. 2.41. Екран для обчислення фондоємності

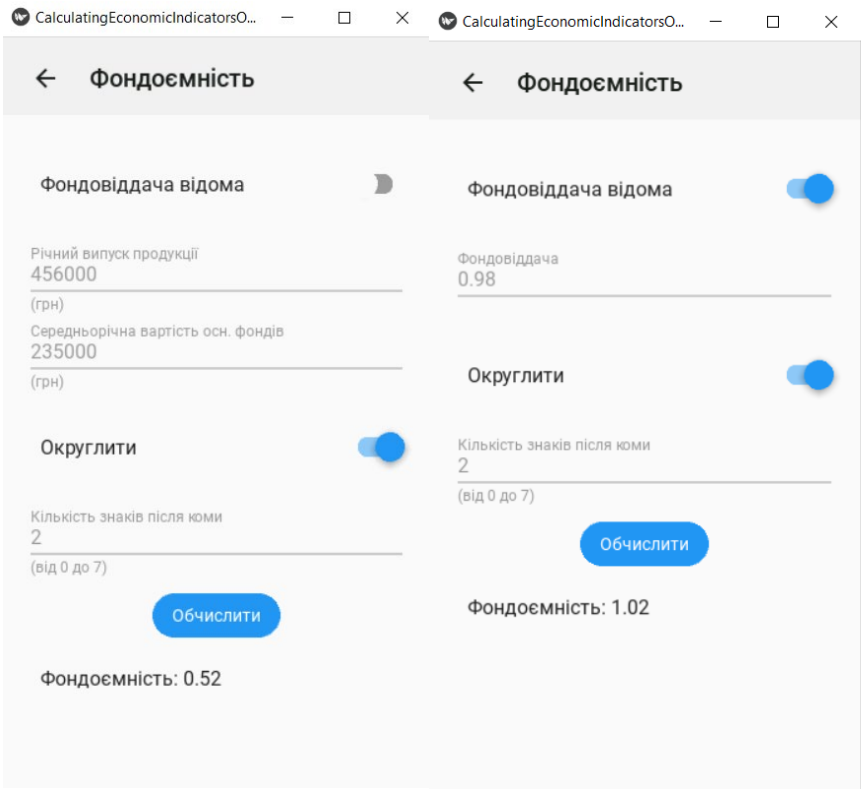

Рис. 2.42. Приклад обчислення фондоємності

Екран для обчислення фондоозброєності, а також приклад виконання розрахунку, зображено на рис. 2.43. Екран для розрахунку інтегрального коефіцієнта використання, а також приклад виконання обчислення, зображено на рис. 2.44.

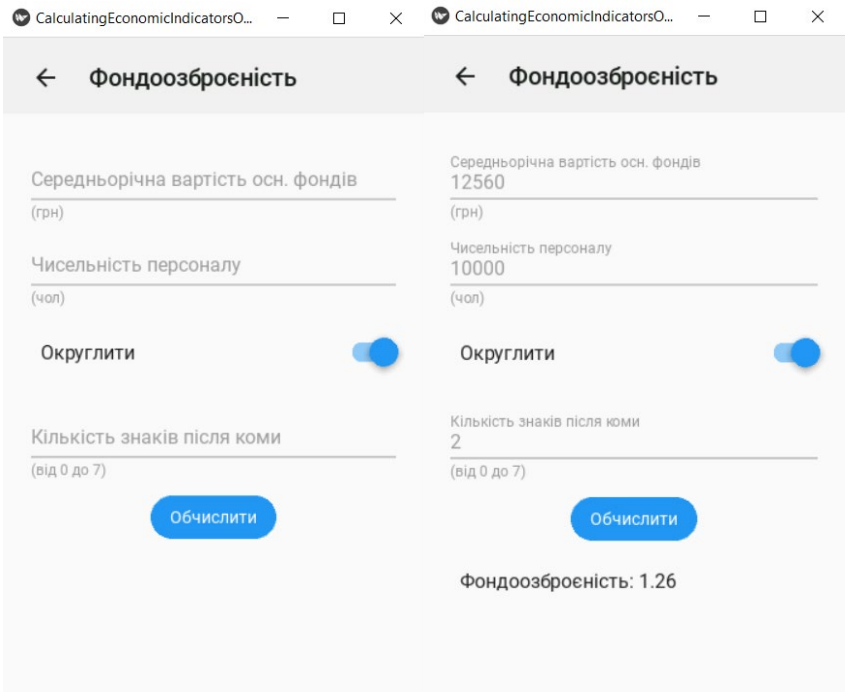

Рис. 2.43. Екран для розрахунку фондоозброєності

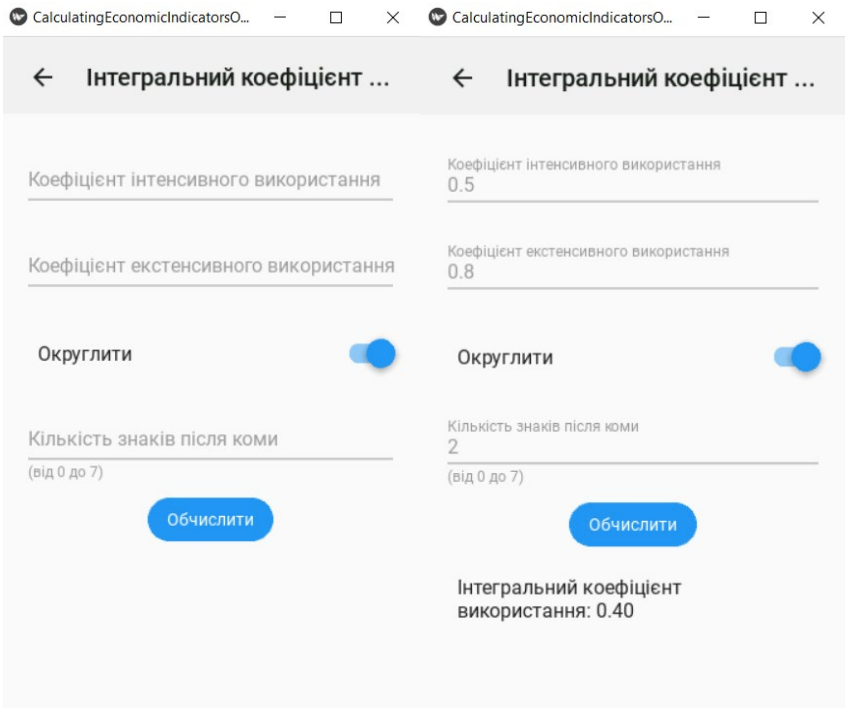

Рис. 2.44. Екран для розрахунку інтегрального коефіцієнта використання

Екран для обчислення показника рентабельності основних фондів, а також приклад виконання розрахунків, представлено на рис. 2.45.

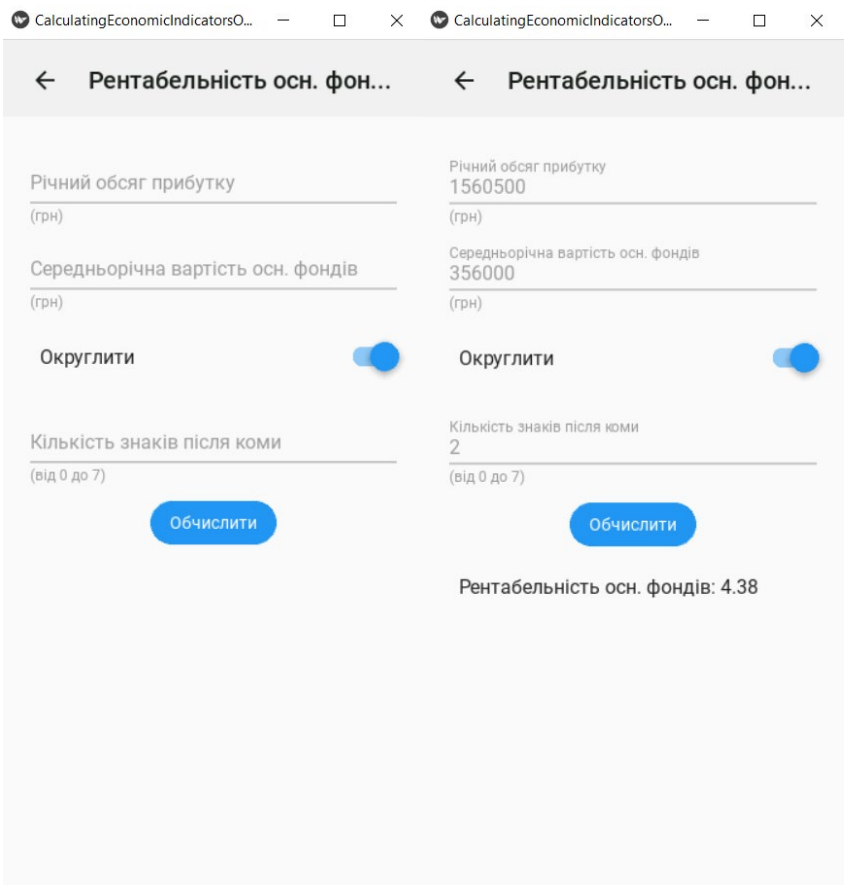

Рис. 2.45. Екран для розрахунку рентабельності основних фондів

В процесі розробки мобільного додатку було використано ресурси [14 - 17].

2.7. Формування виконуваних файлів розробленого ПЗ

Для використання створеного програмного забезпечення на мобільних девайсах з операційною системою Android було сформовано APK-файл, який дозволяє користувачу встановити та використовувати застосунок без необхідності встановлення оболонки та бібліотек Python. Крім того, для роботи з даним ПЗ на настільних пристроях з ОС Windows було створено exe-файл, що надає можливість використовувати програмне забезпечення на пристроях за відсутності інтерпретатора Python.

Для створення APK-файлу використовувалась утиліта Buildozer, а для формування exe-файлу – утиліта pyinstaller. Варто зазначити, що використання утиліти Buildozer для формування APK-файлу можливе лише на операційній системі Linux, а для створення exe-файлу треба використовувати ОС Windows.

Таким чином, для використання Buildozer було завантажено віртуальну машину з ОС Linux, на яку було встановлено утиліту Buildozer, середовище розробки PyCharm, а також фреймворк Kivy та бібліотеку KivyMD. Крім того, було завантажено вихідний код ПЗ, а також сформовано файл buildozer.spec, який містить необхідні параметри налаштувань для створення APK-файлу. Після цього було сформовано APK-файл.

Для створення exe-файлу на операційну систему Windows було встановлено утиліту pyinstaller. Після цього було згенеровано та налаштовано файл специфікації з розширенням .spec. В результаті було створено виконуваний файл з розширенням .exe.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 2278;

- коефіцієнт складності програми - 1,4;

- коефіцієнт корекції програми в ході розробки - 0,1;

- годинна заробітна плата програміста - 85 грн/год;

- коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі - 1,3;

- коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності - 1,2;

- вартість машино-години ЕОМ - 15 грн/год.

Оскільки дане ПЗ не потребувало великої кількості доробок, то коефіцієнт кореляції програми в ході розробки приймається за 0,1.

Оскільки середня заробітна плата Python-розробника становить 20000 грн за [18], то, враховуючи обсяг досвіду, для розробки даного застосунку приймається заробітна плата в обсязі 15000 грн. Таким чином, якщо врахувати, що в місяці в середньому 22 робочих дні по 8 годин, то годинна заробітна плата складає 85 грн.

Оскільки за годину комп’ютер витрачає 0,55 кВт (відповідно до [19]), а вартість електроенергії за кВт/год складає 1,68 грн [20], то за годину комп’ютер споживає електроенергії на $0,55 \cdot 1,68 = 1$ грн. Якщо додати інші витрати на забезпечення роботи комп’ютера, то вартість машино-години ЕОМ становить приблизно 15 грн/год.

Для визначення трудомісткості розробки ПЗ необхідно скористатись формулою (3.1).

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o – витрати праці на підготовку та опис поставленої задачі,

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування за блок-схемою,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Витрати праці на підготовку та опис поставленої задачі для даного ПЗ приймаються за 50 людино-годин.

Для того, щоб розрахувати складові витрати праці, треба обчислити умовне число операторів у даному ПЗ за формулою (3.2).

$$Q = q \cdot C \cdot (1 + p), \text{ людино-годин,} \quad (3.2)$$

де q – передбачуване число операторів,

C - коефіцієнт складності програми (від 1,25 до 2),

p - коефіцієнт кореляції програми в ході розробки.

Обчислення умовного числа операторів представлено у (3.3).

$$Q = 2278 \cdot 1{,}4 \cdot (1 + 0{,}1) = 3508 \text{ людино-годин.} \quad (3.3)$$

Для розрахунку витрат праці на дослідження алгоритму рішення задачі треба використати формулу (3.4).

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \quad (3.4)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (від 1,2 до 1,5),

k - коефіцієнт кваліфікації програміста за стажем роботи за спеціальністю.

Розрахунок витрат праці на дослідження алгоритму рішення задачі наведено у (3.5).

$$t_u = \frac{3508 \cdot 1{,}3}{80 \cdot 1{,}2} = 48 \text{ людино-годин.} \quad (3.5)$$

Витрати праці на розробку блок-схеми алгоритму розраховуються за формулою (3.6).

$$t_a = \frac{Q}{(20..25) \cdot k}, \text{ людино-годин.} \quad (3.6)$$

Таким чином, обчислення витрат праці на розробку блок-схеми алгоритму представлено у (3.7).

$$t_a = \frac{3508}{22 \cdot 1{,}2} = 133 \text{ людино-години.} \quad (3.7)$$

Для обчислення витрат праці на програмування за блок-схемою необхідно використати формулу (3.8).

$$t_n = \frac{Q}{(20..25) \cdot k}, \text{ людино-годин.} \quad (3.8)$$

Таким чином, розрахунок витрат праці на програмування за блок-схемою алгоритму наведено у (3.9).

$$t_n = \frac{3508}{22 \cdot 1{,}2} = 133 \text{ людино-години.} \quad (3.9)$$

Розрахунок витрат праці на налагодження програми на ЕОМ ведеться за формулою (3.10).

$$t_{отл} = \frac{Q}{(4..5) \cdot k}, \text{ людино-годин.} \quad (3.10)$$

Отже, обчислення витрат праці на налагодження програми на ЕОМ представлено у (3.11).

$$t_{отл} = \frac{3508}{4{,}5 \cdot 1{,}2} = 650 \text{ людино-годин.} \quad (3.11)$$

Обчислення витрат праці на підготовку документації здійснюється за формулою (3.12).

$$t_{д} = t_{др} + t_{до}, \text{ людино-годин,} \quad (3.12)$$

де $t_{др}$ – трудомісткість підготовки матеріалів та рукописів,

$t_{до}$ - трудомісткість редагування, друку та оформлення документації.

Трудомісткість підготовки матеріалів та рукописів обчислюється за формулою (3.13).

$$t_{др} = \frac{Q}{(15..20) \cdot k}, \text{ людино-годин.} \quad (3.13)$$

Розрахунок трудомісткості підготовки матеріалів та рукописів представлено у (3.14).

$$t_{др} = \frac{3508}{17 \cdot 1{,}2} = 172 \text{ людино-години.} \quad (3.14)$$

Трудомісткість редагування, друку та оформлення документації розраховується за формулою (3.15).

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.15)$$

Обчислення трудомісткості редагування, друку та оформлення документації наведено у (3.16).

$$t_{до} = 0{,}75 \cdot 172 = 129 \text{ людино-годин.} \quad (3.16)$$

Таким чином, витрати праці на підготовку документації складають (3.17).

$$t_{д} = 172 + 129 = 301 \text{ людино-година.} \quad (3.17)$$

Отже, трудомісткість розробки програмного забезпечення становить (3.18).

$$t = 50 + 48 + 133 + 133 + 650 + 301 = 1315 \text{ людино-годин.} \quad (3.18)$$

3.2. Розрахунок витрат на створення програми

Розрахунок витрат на створення ПЗ здійснюється за формулою (3.19).

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.19)$$

де $З_{ЗП}$ – витрати на заробітну плату виконавця,

$З_{МВ}$ - витрати машинного часу на налагодження програми на ЕОМ.

Для обчислення заробітної плати виконавця необхідно використати формулу (3.20).

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.20)$$

де t – загальна трудомісткість (людино-годин),

$С_{ПР}$ – середня годинна заробітна плата виконавця (грн/год).

Розрахунок заробітної плати наведено у (3.21).

$$З_{ЗП} = 1315 \cdot 85 = 111775 \text{ грн.} \quad (3.21)$$

Для розрахунку вартості машинного часу на налагодження програми на ЕОМ використовується формула (3.22).

$$З_{МВ} = t_{отл} \cdot C_M, \text{ грн,} \quad (3.22)$$

де $t_{отл}$ – трудомісткість налагодження програми,

C_M – вартість машино-години ЕОМ (грн/год).

Таким чином, обчислення вартості машинного часу на налагодження програми на ЕОМ представлено у (3.23).

$$З_{МВ} = 650 \cdot 15 = 9750 \text{ грн.} \quad (3.23)$$

Обчислення витрат на створення ПЗ наведено у (3.24).

$$К_{ПО} = 111775 + 9750 = 121525 \text{ грн.} \quad (3.24)$$

Розрахунок очікуваного періоду створення програмного забезпечення здійснюється за формулою (3.25).

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.,} \quad (3.25)$$

де B_k – число виконавців,

F_p – місячний фонд робочого часу (год).

Оскільки в тижні 5 робочих днів по 8 годин, то місячний фонд робочого часу становить 176 годин.

Обчислення очікуваного періоду створення ПЗ представлено у (3.26).

$$T = \frac{1315}{1 \cdot 176} = 7 \text{ місяців.} \quad (3.26)$$

Таким чином, трудомісткість розроблюваного мобільного додатку складає 1315 людино-год., вартість робіт для розробки ПЗ становить 121525 грн, а час на створення – 7 місяців.

ВИСНОВКИ

Для успішного функціонування підприємства в сучасному світі необхідно здійснювати обчислення та проводити аналіз економічних показників діяльності підприємства.

Призначення даного ПЗ полягає в тому, щоб надати користувачам функціонал для розрахунку амортизації та економічних показників. Крім того, мобільний додаток призначений скоротити витрати часу на виконання обчислень та зменшити ймовірність допускання помилки в процесі здійснення розрахунків.

Метою даної роботи є розробка програмного забезпечення для мобільних пристроїв, що надає функціонал для проведення розрахунку економічних показників підприємства та обчислення амортизації, а також для скорочення витрат часу на розрахунок амортизації та економічних показників.

Актуальність мобільного додатку полягає в необхідності виконання обчислень та аналізу економічних показників підприємства для його успішної роботи, а також у відсутності альтернативи такого ПЗ для мобільних пристроїв з різними ОС, яке функціонує без підключення до мережі Інтернет.

В результаті виконання роботи було розроблено мобільний додаток, який дозволяє:

- обчислити основні економічні показники (середньорічну вартість основних фондів, коефіцієнти вибуття та відновлення, коефіцієнт зносу, номінальний та дійсний фонди, коефіцієнти екстенсивного та інтенсивного використання, фондовіддачу, фондоємність, фондоозброєність, інтегральний коефіцієнт використання, а також рентабельність основних фондів);

- виконати обчислення амортизації прямолінійним методом, методом зменшення залишкової вартості, методом прискореного зменшення залишкової вартості та кумулятивним методом.

Розроблене ПЗ знижує витрати часу на виконання обчислень амортизації та економічних показників підприємства, зменшує ризик здійснення помилки

при виконанні обчислень, а також функціонує на девайсах з різними ОС без необхідності підключення до мережі Інтернет.

Програмне забезпечення розроблене з використанням мови програмування Python та фреймворку Kivy. Крім того, в процесі розробки додатку було використано середовище програмування PyCharm Community Edition 2022.3.2, а також набір офісних програм Microsoft Office.

Створений мобільний додаток призначений для використання у сфері економіки підприємства.

Трудомісткість ПЗ становить 1315 людино-годин, вартість робіт для здійснення розробки додатку складає 121525 грн, а час на створення – 7 місяців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Економіка як сфера діяльності людей, галузь науки та навчальна дисципліна. URL: http://www.ekonomikam.com/ecfins-1601-1.html. Дата звернення: 17.04.2023.
2. Бойчик І.М. Економіка підприємства: підручник. Київ: Кондор, 2016. 378 с.
3. Нарахування амортизації основних засобів в бухгалтерському обліку. URL: https://buhgalter911.com/uk/programmy/calculators/narahuvanie-amortizacii-osnovnyh-zasobiv-v-buh-obliku.html. Дата звернення: 18.04.2023.
4. GlobalStats. Desktop vs Mobile vs Tablet Market Share Worldwide Mar 2022 - Apr 2023. URL: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide. Дата звернення: 25.04.2023.
5. Які технології використовуються для розробки мобільних додатків. URL: https://smart-solutions.com.ua/archives/632. Дата звернення: 25.04.2023.
6. Obvious advantages of cross-platform development. URL: https://linkupst.com/blog/advantages-of-cross-platform-development/. Дата звернення: 25.04.2023.
7. Які існують методи амортизації для основних засобів згідно НПСБО. URL: https://www.buh24.com.ua/knowledge-base/yaki-isnuyut-metodi-amortizatsiyi-dlya-osnovnih-zasobiv-zgidno-npsbo/. Дата звернення: 25.04.2023.
8. Методи амортизації основних засобів. URL: https://zakon.help/article/metodi-amortizacii-osnovnih-zasobiv-oz. Дата звернення: 25.04.2023.
9. Метод прискореного зменшення залишкової вартості. URL: https://www.buhoblik.org.ua/uchet/uchet-osnovnyx-sredstv/1064-metod-priskorenogo-zmenshennya-zalishkovoi-vartosti.html. Дата звернення: 25.04.2023.
10. Офіційний сайт Python. URL: https://www.python.org/about/. Дата звернення: 25.04.2023.

11. Lutz M. Learning Python, 4th Edition / M. Lutz. - O'Reilly Media, 2009. – 1216 p.

12. Офіційний сайт Kivy. URL: https://kivy.org. Дата звернення: 25.04.2023.

13. Phillips D. Creating Apps in Kivy Mobile with Python / D. Phillips. - O'Reilly Media, 2014. – 188 p.

14. Офіційна документація Kivy. URL: https://kivy.org/doc/stable/. Дата звернення: 25.04.2023.

15. Офіційна документація KivyMD. URL: https://kivymd.readthedocs.io/en/1.1.1/components/. Дата звернення: 25.04.2023.

16. Python. How to use Multiple kv files in kivy. URL: https://www.geeksforgeeks.org/python-how-to-use-multiple-kv-files-in-kivy/. Дата звернення: 25.04.2023.

17. Kivy: Use get_screen() to Access Objects from Other Screens. URL: https://medium.com/nerd-for-tech/kivy-use-get-screen-to-access-objects-from-other-screens-8d4d6f288f3. Дата звернення: 25.04.2023.

18. Python-розробник: середня зарплата в Україні. URL: https://www.work.ua/salary-Python-розробник/?count=by-resumes. Дата звернення: 26.05.2023.

19. Скільки електроенергії споживають ваші побутові прилади – підрахунки. URL: https://3oko.com.ua/skil-ky-elektroenerhii-spozhyvaiut-vashi-pobutovi-prylady-pidrakhunky/. Дата звернення: 26.05.2023.

20. Чинний тариф на електроенергію для населення. URL: https://www.ez.rv.ua/chynnyj-taryf-na-elektroenergiyu-dlya-naselennya-prodovzheno-do-31-05-2023/. Дата звернення: 26.05.2023.

ДОДАТОК А

ТЕКСТ ВИХІДНОГО КОДУ ПРОГРАМИ

main.py

```
from kivy.lang import Builder
from kivymd.app import MDApp
from kivy.uix.screenmanager import ScreenManager, Screen
from amortization_straight_method_screen import AmortizationStraightMethodScreen
from residual_value_reduction_method_screen import ResidualValueReductionMethodScreen
from residual_value_accelerated_reduction_method_screen import
ResidualValueAcceleratedReductionMethodScreen
from cumulative_method_screen import CumulativeMethodScreen
from average_annual_cost_screen import AverageAnnualCostScreen
from dropout_coefficient_screen import DropoutCoefficientScreen
from recovery_coefficient_screen import RecoveryCoefficientScreen
from wear_rate_screen import WearRateScreen
from nominal_fund_screen import NominalFundScreen
from effective_fund_screen import EffectiveFundScreen
from extensive_use_coefficient_screen import ExtensiveUseCoefficientScreen
from intensive_use_coefficient_screen import IntensiveUseCoefficientScreen
from fund_return_screen import FundReturnScreen
from fund_capacity_screen import FundCapacityScreen
from fund_availability_screen import FundAvailabilityScreen
from equipment_utilization_integral_coefficient import EquipmentUtilizationIntegralCoefficientScreen
from main_assets_profitability_screen import MainAssetsProfitabilityScreen

# Підключення файлів макетів екранів
Builder.load_file('main_screen.kv')
Builder.load_file('amortization_straight_method_screen.kv')
Builder.load_file('residual_value_reduction_method_screen.kv')
Builder.load_file('residual_value_accelerated_reduction_method_screen.kv')
Builder.load_file('cumulative_method_screen.kv')
Builder.load_file('average_annual_cost_screen.kv')
Builder.load_file('dropout_coefficient_screen.kv')
Builder.load_file('recovery_coefficient_screen.kv')
Builder.load_file('wear_rate_screen.kv')
Builder.load_file('nominal_fund_screen.kv')
Builder.load_file('effective_fund_screen.kv')
Builder.load_file('extensive_use_coefficient_screen.kv')
Builder.load_file('intensive_use_coefficient_screen.kv')
Builder.load_file('fund_return_screen.kv')
Builder.load_file('fund_capacity_screen.kv')
Builder.load_file('fund_availability_screen.kv')
Builder.load_file('equipment_utilization_integral_coefficient.kv')
Builder.load_file('main_assets_profitability_screen.kv')


# Клас головного екрану
class MainScreen(Screen):
    pass


# Головний клас додатку
class CalculatingEconomicIndicatorsOfEnterpriseApp(MDApp):
    def build(self):
        # Налаштування менеджеру екранів
        screen_manager = ScreenManager()
        screen_manager.add_widget(MainScreen(name='MainScreen'))
        screen_manager.add_widget(
            AmortizationStraightMethodScreen(name='AmortizationStraightMethodScreen')
```

```
        )
        screen_manager.add_widget(
            ResidualValueReductionMethodScreen(name='ResidualValueReductionMethodScreen')
        )
        screen_manager.add_widget(
            ResidualValueAcceleratedReductionMethodScreen(
                name='ResidualValueAcceleratedReductionMethodScreen'
            )
        )
        screen_manager.add_widget(CumulativeMethodScreen(name='CumulativeMethodScreen'))
        screen_manager.add_widget(AverageAnnualCostScreen(name='AverageAnnualCostScreen'))
        screen_manager.add_widget(DropoutCoefficientScreen(name='DropoutCoefficientScreen'))
        screen_manager.add_widget(RecoveryCoefficientScreen(name='RecoveryCoefficientScreen'))
        screen_manager.add_widget(WearRateScreen(name='WearRateScreen'))
        screen_manager.add_widget(NominalFundScreen(name='NominalFundScreen'))
        screen_manager.add_widget(EffectiveFundScreen(name='EffectiveFundScreen'))
        screen_manager.add_widget(
            ExtensiveUseCoefficientScreen(name='ExtensiveUseCoefficientScreen')
        )
        screen_manager.add_widget(
            IntensiveUseCoefficientScreen(name='IntensiveUseCoefficientScreen')
        )
        screen_manager.add_widget(FundReturnScreen(name='FundReturnScreen'))
        screen_manager.add_widget(FundCapacityScreen(name='FundCapacityScreen'))
        screen_manager.add_widget(FundAvailabilityScreen(name='FundAvailabilityScreen'))
        screen_manager.add_widget(
            EquipmentUtilizationIntegralCoefficientScreen(
                name='EquipmentUtilizationIntegralCoefficientScreen'
            )
        )
        screen_manager.add_widget(MainAssetsProfitabilityScreen(name='MainAssetsProfitabilityScreen'))
        return screen_manager


if __name__ == '__main__':
    CalculatingEconomicIndicatorsOfEnterpriseApp().run()
```

amortization_straight_method_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField
from kivymd.uix.boxlayout import MDBoxLayout
from kivymd.uix.label import MDLabel

# Клас екрану для обчислення амортизації прямолінійним методом
class AmortizationStraightMethodScreen(Screen):
    dialog = None # змінна, що буде вказувати на діалогове вікно
    initial_value_text_field = None # змінна, що буде вказувати на текстове поле для первісної вартості
    costs_label_layout = None # змінна, що буде вказувати на блок з написом "Витрати на:"
    equipment_purchase_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат
на придбання устаткування
    transportation_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на
транспортування
    installation_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на
установку
    construction_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на
будівельні роботи
    round_up_to_text_field = None # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що викликається при ініціалізації
    def __init__(self, **kwargs):
```

```
        super().__init__(**kwargs)
        # Відображення текстового поля для первісної вартості
        self.show_initial_value_text_field()

    # Метод, що відображає текстове поле для первісної вартості
    def show_initial_value_text_field(self):
        # Створення текстового поля для первісної вартості
        self.initial_value_text_field = MDTextField(
            hint_text="Первісна вартість",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        # Відображення текстового поля для первісної вартості
        self.ids.initial_value_layout.add_widget(self.initial_value_text_field)

    # Метод, що розраховує амортизацію прямолінійним методом
    def calculate(self):
        # Якщо опція "Первісна вартість відома" обрана
        if self.ids.initial_value_is_known_switch.active:
            # Отримання введених користувачем даних з полів для введення
            initial_value_input = self.initial_value_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [initial_value_input, liquidation_value_input,
term_of_useful_use_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
            elif self.values_start_with('0', [initial_value_input, liquidation_value_input,
term_of_useful_use_input]):
                self.show_dialog("Значення мають бути більше за нуль!")
            # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то
відобразити діалогове вікно
            elif self.values_start_with('.', [initial_value_input, liquidation_value_input]):
                self.show_dialog("Значення не може починатись з крапки!")
            else:
                # Перетворення значень з полів для введення з рядків на числа
                initial_value = float(initial_value_input)
                liquidation_value = float(liquidation_value_input)
                # Обчислення та відображення результату
                self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)
        else: # якщо опція "Первісна вартість відома" не обрана
            # Отримання введених користувачем даних з полів для введення
            equipment_purchase_costs_input = self.equipment_purchase_costs_text_field.text
            transportation_costs_input = self.transportation_costs_text_field.text
            installation_costs_input = self.installation_costs_text_field.text
            construction_costs_input = self.construction_costs_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input,
                term_of_useful_use_input
```

```
            ]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input,
                term_of_useful_use_input
            ]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то
відобразити діалогове вікно
            elif self.values_start_with('.', [
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input
            ]):
                self.show_dialog("Значення не може починатись з крапки!")
            # Якщо значення витрат на придбання устаткування починається з '0', то відобразити діалогове
вікно
            elif equipment_purchase_costs_input[0] == '0':
                self.show_dialog("Значення витрат на придбання устаткування має бути більше за нуль!")
            # Якщо значення ліквідаційної вартості починається з '0', то відобразити діалогове вікно
            elif liquidation_value_input[0] == '0':
                self.show_dialog("Значення ліквідаційної вартості має бути більше за нуль!")
            # Якщо значення терміну корисного використання починається з '0', то відобразити діалогове
вікно
            elif term_of_useful_use_input[0] == '0':
                self.show_dialog("Значення терміну корисного використання має бути більше за нуль!")
            else:
                # Обчислення первісної вартості та перетворення значень з полів для введення з рядків на
числа
                initial_value = float(equipment_purchase_costs_input) + float(transportation_costs_input) +
float(installation_costs_input) + float(construction_costs_input)
                liquidation_value = float(liquidation_value_input)
                # Обчислення та відображення результату
                self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)

    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message, # текст діалогового вікна
            buttons=[
                MDFlatButton( # кнопка діалогового вікна
```

```
                text="OK",
                theme_text_color="Custom",
                text_color="#1987ff",
                on_release=lambda _: self.dialog.dismiss() # закриття діалогового вікна при натисканні кнопки "OK"
            )
        ]
    )
    self.dialog.open() # відкриття діалогового вікна

# Метод, що обчислює та відображає результат
def fetch_result(self, initial_value, liquidation_value, term_of_useful_use_input):
    # Якщо ліквідаційна вартість не є меншою ніж первісна, то відобразити діалогове вікно
    if liquidation_value >= initial_value:
        self.show_dialog("Ліквідаційна вартість має бути меншою ніж первісна!")
    else:
        # Якщо опція округлення обрана
        if self.ids.round_switch.active:
            round_up_to_input = self.round_up_to_text_field.text # кількість знаків після коми, введена користувачем
            # Якщо поле порожнє, то відобразити діалогове вікно
            if round_up_to_input == "":
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо значення починається з '-', то відобразити діалогове вікно
            elif round_up_to_input[0] == '-':
                self. show_dialog("Значення мають бути додатні!")
            # Якщо значення більше 7, то відобразити діалогове вікно
            elif int(round_up_to_input) > 7:
                self.show_dialog("Значення має бути не більше 7!")
            else:
                # Розрахунок річної та місячної суми амортизації
                annual_depreciation, monthly_depreciation = self.calculate_result(initial_value, liquidation_value, term_of_useful_use_input)
                # Округлення річної та місячної суми амортизації
                annual_depreciation = round(Decimal(str(annual_depreciation)), int(round_up_to_input))
                monthly_depreciation = round(Decimal(str(monthly_depreciation)), int(round_up_to_input))
                # Відображення річної та місячної суми амортизації
                self.show_result(annual_depreciation, monthly_depreciation)
        else:
            # Розрахунок річної та місячної суми амортизації
            annual_depreciation, monthly_depreciation = self.calculate_result(initial_value, liquidation_value, term_of_useful_use_input)
            # Відображення річної та місячної суми амортизації
            self.show_result(annual_depreciation, monthly_depreciation)

# Метод, що виконує обчислення та повертає результат
def calculate_result(self, initial_value, liquidation_value, term_of_useful_use):
    annual_depreciation = (initial_value - liquidation_value) / int(term_of_useful_use)
    monthly_depreciation = annual_depreciation / 12
    return annual_depreciation, monthly_depreciation

# Метод, що відображає результат обчислень
def show_result(self, annual_depreciation, monthly_depreciation):
    self.ids.annual_depreciation.text = "Річна сума амортизації: " + str(annual_depreciation) + " грн"
    self.ids.monthly_depreciation.text = "Місячна сума амортизації: " + str(monthly_depreciation) + " грн"

# Метод, що повертає на попередній екран
def navigate_back(self):
    self.manager.transition.direction = 'right'
    self.manager.current = 'MainScreen'

# Метод, що викликається при зміні стану перемикача для первісної вартості
def on_initial_value_is_known_switch_active(self, checkbox, value):
```

```
        if value: # якщо перемикач ввімкнений
            # Прибирання блоку з написом "Витрати на:"
            self.ids.initial_value_layout.remove_widget(self.costs_label_layout)
            # Прибирання текстових полів для витрат
            self.ids.initial_value_layout.remove_widget(self.equipment_purchase_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.transportation_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.installation_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.construction_costs_text_field)
            # Відображення текстового поля для первісної вартості
            self.show_initial_value_text_field()
        else: # інакше
            # Прибирання текстового поля для первісної вартості з екрану
            self.ids.initial_value_layout.remove_widget(self.initial_value_text_field)

            # Створення та відображення блоку з написом "Витрати на:"
            self.costs_label_layout = MDBoxLayout(orientation="horizontal", adaptive_height=True,
padding="8dp")
            self.costs_label_layout.add_widget(MDLabel(text="Витрати на:"))
            self.ids.initial_value_layout.add_widget(self.costs_label_layout)

            # Створення текстових полів для витрат
            self.equipment_purchase_costs_text_field = MDTextField(
                hint_text="Придбання устаткування",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )
            self.transportation_costs_text_field = MDTextField(
                hint_text="Транспортування",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )
            self.installation_costs_text_field = MDTextField(
                hint_text="Установку",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )
            self.construction_costs_text_field = MDTextField(
                hint_text="Будівельні роботи (для споруд)",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )
            # Відображення текстових полів для витрат
            self.ids.initial_value_layout.add_widget(self.equipment_purchase_costs_text_field)
            self.ids.initial_value_layout.add_widget(self.transportation_costs_text_field)
            self.ids.initial_value_layout.add_widget(self.installation_costs_text_field)
            self.ids.initial_value_layout.add_widget(self.construction_costs_text_field)

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value: # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
```

```
            self.ids.round_layout.adaptive_height = True
        else: # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

average_annual_cost_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField

# Клас екрану для обчислення середньорічної вартості основних фондів
class AverageAnnualCostScreen(Screen):
    dialog = None # змінна, що буде вказувати на діалогове вікно
    year_ending_assets_cost_text_field = None # змінна, що буде вказувати на текстове поле для вартості основних фондів на кінець року
    entered_assets_cost_text_field = None # змінна, що буде вказувати на текстове поле для введених основних фондів
    lost_assets_cost_text_field = None # змінна, що буде вказувати на текстове поле для вибулих основних фондів
    round_up_to_text_field = None # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що викликається при ініціалізації
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        # Відображення текстового поля для вартості основних фондів на кінець року
        self.show_ending_assets_cost_text_field()

    # Метод, що відображає текстове поле для вартості основних фондів на кінець року
    def show_ending_assets_cost_text_field(self):
        # Створення текстового поля для вартості основних фондів на кінець року
        self.year_ending_assets_cost_text_field = MDTextField(
            hint_text="Вартість осн. фондів на кінець року",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        # Відображення текстового поля для вартості основних фондів на кінець року
        self.ids.year_ending_assets_cost_layout.add_widget(self.year_ending_assets_cost_text_field)

    # Метод, що розраховує середньорічну вартість основних фондів
    def calculate(self):
        # Якщо опція "Вартість основних фондів на кінець року відома" обрана
        if self.ids.year_ending_assets_cost_is_known_switch.active:
            # Отримання введених користувачем даних з полів для введення
            year_beginning_assets_cost_input = self.ids.year_beginning_assets_cost.text
            year_ending_assets_cost_input = self.year_ending_assets_cost_text_field.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([year_beginning_assets_cost_input, year_ending_assets_cost_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [year_beginning_assets_cost_input, year_ending_assets_cost_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
            elif self.values_start_with('0', [year_beginning_assets_cost_input, year_ending_assets_cost_input]):
                self.show_dialog("Значення мають бути більше за нуль!")
            # Якщо хоча б одне значення починається з '.', то відобразити діалогове вікно
            elif self.values_start_with('.', [year_beginning_assets_cost_input, year_ending_assets_cost_input]):
                self.show_dialog("Значення не може починатись з крапки!")
            else:
```

```
            # Перетворення значень з полів для введення з рядків на числа
            year_beginning_assets_cost = float(year_beginning_assets_cost_input)
            year_ending_assets_cost = float(year_ending_assets_cost_input)
            # Обчислення та відображення результату
            self.fetch_result(year_beginning_assets_cost, year_ending_assets_cost)
        else: # якщо опція "Вартість основних фондів на кінець року відома" не обрана
            # Отримання введених користувачем даних з полів для введення
            year_beginning_assets_cost_input = self.ids.year_beginning_assets_cost.text
            entered_assets_cost_input = self.entered_assets_cost_text_field.text
            lost_assets_cost_input = self.lost_assets_cost_text_field.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([year_beginning_assets_cost_input, entered_assets_cost_input,
lost_assets_cost_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [year_beginning_assets_cost_input, entered_assets_cost_input,
lost_assets_cost_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо вартість основних фондів на початок року починається з '0', то відобразити діалогове
вікно
            elif year_beginning_assets_cost_input[0] == '0':
                self.show_dialog("Вартість основних фондів на початок року має бути більше за нуль!")
            # Якщо хоча б одне значення починається з '.', то відобразити діалогове вікно
            elif self.values_start_with('.', [year_beginning_assets_cost_input, entered_assets_cost_input,
lost_assets_cost_input]):
                self.show_dialog("Значення не може починатись з крапки!")
            else:
                # Перетворення значень з полів для введення з рядків на числа та обчислення вартості
основних фондів на кінець року
                year_beginning_assets_cost = float(year_beginning_assets_cost_input)
                year_ending_assets_cost = year_beginning_assets_cost + float(entered_assets_cost_input) -
float(lost_assets_cost_input)
                # Обчислення та відображення результату
                self.fetch_result(year_beginning_assets_cost, year_ending_assets_cost)

    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message, # текст діалогового вікна
            buttons=[
                MDFlatButton( # кнопка діалогового вікна
                    text="OK",
                    theme_text_color="Custom",
                    text_color="#1987ff",
                    on_release=lambda _: self.dialog.dismiss() # закриття діалогового вікна при натисканні
кнопки "OK"
                )
```

```
            ]
        )
        self.dialog.open() # відкриття діалогового вікна

    # Метод, що обчислює та відображає результат
    def fetch_result(self, year_beginning_assets_cost, year_ending_assets_cost):
        # Якщо опція округлення обрана
        if self.ids.round_switch.active:
            round_up_to_input = self.round_up_to_text_field.text # кількість знаків після коми, введена користувачем
            # Якщо поле порожнє, то відобразити діалогове вікно
            if round_up_to_input == "":
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо значення починається з '-', то відобразити діалогове вікно
            elif round_up_to_input[0] == '-':
                self. show_dialog("Значення мають бути додатні!")
            # Якщо значення більше 7, то відобразити діалогове вікно
            elif int(round_up_to_input) > 7:
                self.show_dialog("Значення має бути не більше 7!")
            else:
                # Обчислення та відображення результату з округленням
                self.show_result(year_beginning_assets_cost, year_ending_assets_cost, int(round_up_to_input))
        else:
            # Обчислення та відображення результату
            self.show_result(year_beginning_assets_cost, year_ending_assets_cost, None)

    # Метод, що виконує обчислення та відображає результат обчислень
    def show_result(self, year_beginning_assets_cost, year_ending_assets_cost, round_up_to):
        # Обчислення середньорічної вартості основних фондів
        average_annual_cost = (year_beginning_assets_cost + year_ending_assets_cost) / 2

        # Округлення, якщо така опція обрана користувачем
        if round_up_to != None:
            average_annual_cost = round(Decimal(str(average_annual_cost)), round_up_to)

        # Відображення результату обчислень
        self.ids.average_annual_cost.text = "Середньорічна вартість основних фондів: " + str(average_annual_cost) + " грн"

    # Метод, що повертає на попередній екран
    def navigate_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'MainScreen'

    # Метод, що викликається при зміні стану перемикача для вартості основних фондів на кінець року
    def on_year_ending_assets_cost_is_known_switch_active(self, checkbox, value):
        if value: # якщо перемикач ввімкнений
            # Прибирання текстових полів для введених та вибулих фондів
            self.ids.year_ending_assets_cost_layout.remove_widget(self.entered_assets_cost_text_field)
            self.ids.year_ending_assets_cost_layout.remove_widget(self.lost_assets_cost_text_field)
            # Відображення текстового поля для вартості основних фондів на кінець року
            self.show_ending_assets_cost_text_field()
        else: # інакше
            # Прибирання текстового поля для вартості основних фондів на кінець року з екрану
            self.ids.year_ending_assets_cost_layout.remove_widget(self.year_ending_assets_cost_text_field)

            # Створення текстових полів для введених та вибулих фондів
            self.entered_assets_cost_text_field = MDTextField(
                hint_text="Вартість введених фондів",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )
```

```
            self.lost_assets_cost_text_field = MDTextField(
                hint_text="Вартість вибулих фондів",
                input_filter="float",
                helper_text="(грн)",
                helper_text_mode="persistent"
            )

            # Відображення текстових полів для введених та вибулих фондів
            self.ids.year_ending_assets_cost_layout.add_widget(self.entered_assets_cost_text_field)
            self.ids.year_ending_assets_cost_layout.add_widget(self.lost_assets_cost_text_field)

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value: # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = True
        else: # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

cumulative_method_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField
from kivymd.uix.boxlayout import MDBoxLayout
from kivymd.uix.label import MDLabel

# Клас екрану для обчислення амортизації кумулятивним методом
class CumulativeMethodScreen(Screen):
    dialog = None # змінна, що буде вказувати на діалогове вікно
    initial_value_text_field = None # змінна, що буде вказувати на текстове поле для первісної вартості
    costs_label_layout = None # змінна, що буде вказувати на блок з написом "Витрати на:"
    equipment_purchase_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на придбання устаткування
    transportation_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на транспортування
    installation_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на установку
    construction_costs_text_field = None # змінна, що буде вказувати на текстове поле для витрат на будівельні роботи
    round_up_to_text_field = None # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що викликається при ініціалізації
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        # Відображення текстового поля для первісної вартості
        self.show_initial_value_text_field()

    # Метод, що відображає текстове поле для первісної вартості
    def show_initial_value_text_field(self):
        # Створення текстового поля для первісної вартості
        self.initial_value_text_field = MDTextField(
```

```
            hint_text="Первісна вартість",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        # Відображення текстового поля для первісної вартості
        self.ids.initial_value_layout.add_widget(self.initial_value_text_field)

    # Метод, що розраховує амортизацію кумулятивним методом
    def calculate(self):
        # Якщо опція "Первісна вартість відома" обрана
        if self.ids.initial_value_is_known_switch.active:
            # Отримання введених користувачем даних з полів для введення
            initial_value_input = self.initial_value_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([initial_value_input, liquidation_value_input, term_of_useful_use_input]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [initial_value_input, liquidation_value_input,
term_of_useful_use_input]):
                self.show_dialog("Значення мають бути додатні!")
            # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
            elif self.values_start_with('0', [initial_value_input, liquidation_value_input,
term_of_useful_use_input]):
                self.show_dialog("Значення мають бути більше за нуль!")
            # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то
відобразити діалогове вікно
            elif self.values_start_with('.', [initial_value_input, liquidation_value_input]):
                self.show_dialog("Значення не може починатись з крапки!")
            else:
                # Перетворення значень з полів для введення з рядків на числа
                initial_value = float(initial_value_input)
                liquidation_value = float(liquidation_value_input)
                # Обчислення та відображення результату
                self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)
        else: # якщо опція "Первісна вартість відома" не обрана
            # Отримання введених користувачем даних з полів для введення
            equipment_purchase_costs_input = self.equipment_purchase_costs_text_field.text
            transportation_costs_input = self.transportation_costs_text_field.text
            installation_costs_input = self.installation_costs_text_field.text
            construction_costs_input = self.construction_costs_text_field.text
            liquidation_value_input = self.ids.liquidation_value.text
            term_of_useful_use_input = self.ids.term_of_useful_use.text

            # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
            if self.values_are_empty([
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
                liquidation_value_input,
                term_of_useful_use_input
            ]):
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
            elif self.values_start_with('-', [
                equipment_purchase_costs_input,
                transportation_costs_input,
                installation_costs_input,
                construction_costs_input,
```

```
            liquidation_value_input,
            term_of_useful_use_input
        ]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення з полів для введення десяткового числа починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [
            equipment_purchase_costs_input,
            transportation_costs_input,
            installation_costs_input,
            construction_costs_input,
            liquidation_value_input
        ]):
            self.show_dialog("Значення не може починатись з крапки!")
        # Якщо значення витрат на придбання устаткування починається з '0', то відобразити діалогове вікно
        elif equipment_purchase_costs_input[0] == '0':
            self.show_dialog("Значення витрат на придбання устаткування має бути більше за нуль!")
        # Якщо значення ліквідаційної вартості починається з '0', то відобразити діалогове вікно
        elif liquidation_value_input[0] == '0':
            self.show_dialog("Значення ліквідаційної вартості має бути більше за нуль!")
        # Якщо значення терміну корисного використання починається з '0', то відобразити діалогове вікно
        elif term_of_useful_use_input[0] == '0':
            self.show_dialog("Значення терміну корисного використання має бути більше за нуль!")
        else:
            # Обчислення первісної вартості та перетворення значень з полів для введення з рядків на числа
            initial_value = float(equipment_purchase_costs_input) + float(transportation_costs_input) + float(installation_costs_input) + float(construction_costs_input)
            liquidation_value = float(liquidation_value_input)
            # Обчислення та відображення результату
            self.fetch_result(initial_value, liquidation_value, term_of_useful_use_input)

    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message, # текст діалогового вікна
            buttons=[
                MDFlatButton( # кнопка діалогового вікна
                    text="OK",
                    theme_text_color="Custom",
                    text_color="#1987ff",
                    on_release=lambda _: self.dialog.dismiss() # закриття діалогового вікна при натисканні кнопки "OK"
                )
            ]
        )
```

```
        self.dialog.open() # відкриття діалогового вікна

    # Метод, що обчислює та відображає результат
    def fetch_result(self, initial_value, liquidation_value, term_of_useful_use_input):
        # Якщо ліквідаційна вартість не є меншою ніж первісна, то відобразити діалогове вікно
        if liquidation_value >= initial_value:
            self.show_dialog("Ліквідаційна вартість має бути меншою ніж первісна!")
        else:
            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                round_up_to_input = self.round_up_to_text_field.text # кількість знаків після коми, введена
користувачем
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
                    self.show_dialog("Всі поля мають бути заповнені!")
                # Якщо значення починається з '-', то відобразити діалогове вікно
                elif round_up_to_input[0] == '-':
                    self. show_dialog("Значення мають бути додатні!")
                # Якщо значення більше 7, то відобразити діалогове вікно
                elif int(round_up_to_input) > 7:
                    self.show_dialog("Значення має бути не більше 7!")
                else:
                    # Обчислення та відображення результату
                    self.calculate_result(initial_value, liquidation_value, term_of_useful_use_input,
int(round_up_to_input))
            else:
                # Обчислення та відображення результату
                self.calculate_result(initial_value, liquidation_value, term_of_useful_use_input, None)

    # Метод, що виконує обчислення та відображає результат
    def calculate_result(self, initial_value, liquidation_value, term_of_useful_use, round_up_to):
        years = int(term_of_useful_use) # термін корисного використання
        initial_value_without_liquidation_value = initial_value - liquidation_value # різниця первісної вартості
та ліквідаційної
        residual_value = initial_value # залишкова вартість (спочатку дорівнює первісній вартості)
        previous_year_depreciation = 0.0 # амортизаційні відрахування за попередній рік
        accumulated_depreciation = 0.0 # накопичена амортизація
        years_of_object_use = 0 # кількість років застосування об'єкта
        result_text = "" # текст, що буде містити результати обчислень

        # Розрахунок кількості років застосування об'єкта
        current_years = years # кількість років на поточному кроці
        while current_years >= 1:
            years_of_object_use += current_years
            current_years -= 1

        current_years = years # кількість років на поточному кроці

        for year in range(years): # цикл по всім рокам
            # Обчислення амортизаційних відрахувань
            previous_year_depreciation = initial_value_without_liquidation_value * current_years /
years_of_object_use

            # Обчислення накопиченої амортизації
            if year == 0: # для першого року
                accumulated_depreciation = previous_year_depreciation
            else: # для інших років
                accumulated_depreciation = accumulated_depreciation + previous_year_depreciation

            # Обчислення залишкової вартості
            residual_value = residual_value - previous_year_depreciation
            # Обчислення місячної суми амортизаційних відрахувань
            monthly_depreciation = previous_year_depreciation / 12
```

```
            current_years -= 1 # оновлення кількості років на поточному кроці

            # Додавання результатів обчислень для поточного року до тексту результатів обчислень
            if round_up_to == None: # якщо не треба округляти результат
                result_text += "\nРічна сума амортизаційних відрахувань за "
                result_text += str(year + 1)
                result_text += "-й рік експлуатації: "
                result_text += str(previous_year_depreciation)
                result_text += " грн. Місячна сума амортизаційних відрахувань: "
                result_text += str(monthly_depreciation)
                result_text += " грн. Накопичена амортизація: "
                result_text += str(accumulated_depreciation)
                result_text += " грн. Залишкова вартість: "
                result_text += str(residual_value)
                result_text += " грн.\n"
            else: # якщо треба округляти результат
                result_text += "\nРічна сума амортизаційних відрахувань за "
                result_text += str(year + 1)
                result_text += "-й рік експлуатації: "
                result_text += str(round(Decimal(str(previous_year_depreciation)), round_up_to))
                result_text += " грн. Місячна сума амортизаційних відрахувань: "
                result_text += str(round(Decimal(str(monthly_depreciation)), round_up_to))
                result_text += " грн. Накопичена амортизація: "
                result_text += str(round(Decimal(str(accumulated_depreciation)), round_up_to))
                result_text += " грн. Залишкова вартість: "
                result_text += str(round(Decimal(str(residual_value)), round_up_to))
                result_text += " грн.\n"

        # Відображення результатів обчислень
        self.ids.result.text = result_text

    # Метод, що повертає на попередній екран
    def navigate_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'MainScreen'

    # Метод, що викликається при зміні стану перемикача для первісної вартості
    def on_initial_value_is_known_switch_active(self, checkbox, value):
        if value: # якщо перемикач ввімкнений
            # Прибирання блоку з написом "Витрати на:"
            self.ids.initial_value_layout.remove_widget(self.costs_label_layout)
            # Прибирання текстових полів для витрат
            self.ids.initial_value_layout.remove_widget(self.equipment_purchase_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.transportation_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.installation_costs_text_field)
            self.ids.initial_value_layout.remove_widget(self.construction_costs_text_field)
            # Відображення текстового поля для первісної вартості
            self.show_initial_value_text_field()
        else: # інакше
            # Прибирання текстового поля для первісної вартості з екрану
            self.ids.initial_value_layout.remove_widget(self.initial_value_text_field)

            # Створення та відображення блоку з написом "Витрати на:"
            self.costs_label_layout = MDBoxLayout(orientation="horizontal", adaptive_height=True,
padding="8dp")
            self.costs_label_layout.add_widget(MDLabel(text="Витрати на:"))
            self.ids.initial_value_layout.add_widget(self.costs_label_layout)

            # Створення текстових полів для витрат
            self.equipment_purchase_costs_text_field = MDTextField(
                hint_text="Придбання устаткування",
                input_filter="float",
```

```
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        self.transportation_costs_text_field = MDTextField(
            hint_text="Транспортування",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        self.installation_costs_text_field = MDTextField(
            hint_text="Установку",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        self.construction_costs_text_field = MDTextField(
            hint_text="Будівельні роботи (для споруд)",
            input_filter="float",
            helper_text="(грн)",
            helper_text_mode="persistent"
        )
        # Відображення текстових полів для витрат
        self.ids.initial_value_layout.add_widget(self.equipment_purchase_costs_text_field)
        self.ids.initial_value_layout.add_widget(self.transportation_costs_text_field)
        self.ids.initial_value_layout.add_widget(self.installation_costs_text_field)
        self.ids.initial_value_layout.add_widget(self.construction_costs_text_field)

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value: # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = True
        else: # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

dropout_coefficient_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField

# Клас екрану для обчислення коефіцієнту вибуття
class DropoutCoefficientScreen(Screen):
    dialog = None # змінна, що буде вказувати на діалогове вікно
    round_up_to_text_field = None # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що розраховує коефіцієнт вибуття
    def calculate(self):
        # Отримання введених користувачем даних з полів для введення
        lost_assets_cost_input = self.ids.lost_assets_cost.text
        year_beginning_assets_cost_input = self.ids.year_beginning_assets_cost.text
```

```
        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([lost_assets_cost_input, year_beginning_assets_cost_input]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
        elif self.values_start_with('-', [lost_assets_cost_input, year_beginning_assets_cost_input]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
        elif self.values_start_with('0', [lost_assets_cost_input, year_beginning_assets_cost_input]):
            self.show_dialog("Значення мають бути більше за нуль!")
        # Якщо хоча б одне значення починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [lost_assets_cost_input, year_beginning_assets_cost_input]):
            self.show_dialog("Значення не може починатись з крапки!")
        else:
            # Перетворення значень з полів для введення з рядків на числа
            lost_assets_cost = float(lost_assets_cost_input)
            year_beginning_assets_cost = float(year_beginning_assets_cost_input)

            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                round_up_to_input = self.round_up_to_text_field.text # кількість знаків після коми, введена
користувачем
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
                    self.show_dialog("Всі поля мають бути заповнені!")
                # Якщо значення починається з '-', то відобразити діалогове вікно
                elif round_up_to_input[0] == '-':
                    self. show_dialog("Значення мають бути додатні!")
                # Якщо значення більше 7, то відобразити діалогове вікно
                elif int(round_up_to_input) > 7:
                    self.show_dialog("Значення має бути не більше 7!")
                else:
                    # Обчислення та відображення результату з округленням
                    self.show_result(lost_assets_cost, year_beginning_assets_cost, int(round_up_to_input))
            else:
                # Обчислення та відображення результату
                self.show_result(lost_assets_cost, year_beginning_assets_cost, None)

    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message, # текст діалогового вікна
            buttons=[
                MDFlatButton( # кнопка діалогового вікна
                    text="OK",
                    theme_text_color="Custom",
                    text_color="#1987ff",
                    on_release=lambda _: self.dialog.dismiss() # закриття діалогового вікна при натисканні
кнопки "OK"
```

```
            )
        ]
    )
    self.dialog.open() # відкриття діалогового вікна

# Метод, що виконує обчислення та відображає результат обчислень
def show_result(self, lost_assets_cost, year_beginning_assets_cost, round_up_to):
    # Обчислення результату
    dropout_coefficient = lost_assets_cost / year_beginning_assets_cost

    # Округлення, якщо така опція обрана користувачем
    if round_up_to != None:
        dropout_coefficient = round(Decimal(str(dropout_coefficient)), round_up_to)

    # Відображення результату обчислень
    self.ids.dropout_coefficient.text = "Коефіцієнт вибуття: " + str(dropout_coefficient)

# Метод, що повертає на попередній екран
def navigate_back(self):
    self.manager.transition.direction = 'right'
    self.manager.current = 'MainScreen'

# Метод, що викликається при зміні стану перемикача для округлення
def on_round_switch_active(self, checkbox, value):
    if value: # якщо перемикач ввімкнений
        # Створення текстового поля для округлення
        self.round_up_to_text_field = MDTextField(
            hint_text="Кількість знаків після коми",
            input_filter="int",
            helper_text="(від 0 до 7)",
            helper_text_mode="persistent"
        )
        # Відображення текстового поля для округлення
        self.ids.round_layout.add_widget(self.round_up_to_text_field)
        self.ids.round_layout.adaptive_height = True
    else: # інакше
        # Прибирання текстового поля для округлення з екрану
        self.ids.round_layout.remove_widget(self.round_up_to_text_field)
        self.ids.round_layout.adaptive_height = False
```

effective_fund_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField


# Клас екрану для обчислення дійсного фонду
class EffectiveFundScreen(Screen):
    dialog = None  # змінна, що буде вказувати на діалогове вікно
    round_up_to_text_field = None  # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що розраховує дійсний фонд
    def calculate(self):
        # Отримання введених користувачем даних з полів для введення
        nominal_fund_input = self.ids.nominal_fund.text
        repairs_time_input = self.ids.repairs_time.text

        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([nominal_fund_input, repairs_time_input]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
```

```
    elif self.values_start_with('-', [nominal_fund_input, repairs_time_input]):
        self.show_dialog("Значення мають бути додатні!")
    # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
    elif self.values_start_with('0', [nominal_fund_input, repairs_time_input]):
        self.show_dialog("Значення мають бути більше за нуль!")
    # Якщо хоча б одне значення починається з '.', то відобразити діалогове вікно
    elif self.values_start_with('.', [nominal_fund_input, repairs_time_input]):
        self.show_dialog("Значення не може починатись з крапки!")
    else:
        # Якщо час на ремонти більше ніж 100%, то відобразити діалогове вікно
        if float(repairs_time_input) > 100:
            self.show_dialog(
                "Час на планово-попереджувальні ремонти не може бути більше ніж 100%!"
            )
        else:
            # Обчислення результату
            effective_fund = float(nominal_fund_input) * (1 - (float(repairs_time_input) / 100))

            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                # Кількість знаків після коми, введена користувачем
                round_up_to_input = self.round_up_to_text_field.text
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
                    self.show_dialog("Всі поля мають бути заповнені!")
                # Якщо значення починається з '-', то відобразити діалогове вікно
                elif round_up_to_input[0] == '-':
                    self.show_dialog("Значення мають бути додатні!")
                # Якщо значення більше 7, то відобразити діалогове вікно
                elif int(round_up_to_input) > 7:
                    self.show_dialog("Значення має бути не більше 7!")
                else:
                    # Округлення результату
                    effective_fund = round(Decimal(str(effective_fund)), int(round_up_to_input))
            # Відображення результату
            self.ids.result.text = "Дійсний фонд: " + str(effective_fund) + " год"

# Метод, що перевіряє чи є рядки порожніми
def values_are_empty(self, values):
    for value in values:
        if value == "":
            return True
    return False

# Метод, що перевіряє чи починаються значення з заданого символа
def values_start_with(self, symbol, values):
    for value in values:
        if value[0] == symbol:
            return True
    return False

# Метод, що відображає діалогове вікно
def show_dialog(self, message):
    # Створення діалогового вікна
    self.dialog = MDDialog(
        text=message,  # текст діалогового вікна
        buttons=[
            MDFlatButton(  # кнопка діалогового вікна
                text="OK",
                theme_text_color="Custom",
                text_color="#1987ff",
                on_release=lambda _: self.dialog.dismiss()
                # закриття діалогового вікна при натисканні кнопки "OK"
```

```
            )
        ]
    )
    self.dialog.open()  # відкриття діалогового вікна

# Метод, що повертає на попередній екран
def navigate_back(self):
    self.manager.transition.direction = 'right'
    self.manager.current = 'MainScreen'

# Метод, що викликається при зміні стану перемикача для округлення
def on_round_switch_active(self, checkbox, value):
    if value:  # якщо перемикач ввімкнений
        # Створення текстового поля для округлення
        self.round_up_to_text_field = MDTextField(
            hint_text="Кількість знаків після коми",
            input_filter="int",
            helper_text="(від 0 до 7)",
            helper_text_mode="persistent"
        )
        # Відображення текстового поля для округлення
        self.ids.round_layout.add_widget(self.round_up_to_text_field)
        self.ids.round_layout.adaptive_height = True
    else:  # інакше
        # Прибирання текстового поля для округлення з екрану
        self.ids.round_layout.remove_widget(self.round_up_to_text_field)
        self.ids.round_layout.adaptive_height = False
```

equipment_utilization_integral_coefficient.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField


# Клас екрану для обчислення інтегрального коефіцієнта використання устаткування
class EquipmentUtilizationIntegralCoefficientScreen(Screen):
    dialog = None  # змінна, що буде вказувати на діалогове вікно
    round_up_to_text_field = None  # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що розраховує інтегральний коефіцієнт використання устаткування
    def calculate(self):
        # Отримання введених користувачем даних з полів для введення
        intensive_use_coefficient_input = self.ids.intensive_use_coefficient.text
        extensive_use_coefficient_input = self.ids.extensive_use_coefficient.text

        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([
            intensive_use_coefficient_input,
            extensive_use_coefficient_input
        ]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
        elif self.values_start_with('-', [
            intensive_use_coefficient_input,
            extensive_use_coefficient_input
        ]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [
            intensive_use_coefficient_input,
            extensive_use_coefficient_input
```

```
    ]):
        self.show_dialog("Значення не може починатись з крапки!")
    else:
        # Якщо хоча б одне значення дорівнює 0, то відобразити діалогове вікно
        if float(intensive_use_coefficient_input) == 0 or float(extensive_use_coefficient_input) == 0:
            self.show_dialog("Значення мають бути більше за нуль!")
        else:
            # Обчислення результату
            equipment_utilization_integral_coefficient = \
                float(intensive_use_coefficient_input) * float(extensive_use_coefficient_input)

            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                # Кількість знаків після коми, введена користувачем
                round_up_to_input = self.round_up_to_text_field.text
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
                    self.show_dialog("Всі поля мають бути заповнені!")
                # Якщо значення починається з '-', то відобразити діалогове вікно
                elif round_up_to_input[0] == '-':
                    self.show_dialog("Значення мають бути додатні!")
                # Якщо значення більше 7, то відобразити діалогове вікно
                elif int(round_up_to_input) > 7:
                    self.show_dialog("Значення має бути не більше 7!")
                else:
                    # Округлення результату
                    equipment_utilization_integral_coefficient = round(
                        Decimal(str(equipment_utilization_integral_coefficient)),
                        int(round_up_to_input)
                    )
            # Відображення результату
            self.ids.result.text = "Інтегральний коефіцієнт використання: " + \
                                   str(equipment_utilization_integral_coefficient)

# Метод, що перевіряє чи є рядки порожніми
def values_are_empty(self, values):
    for value in values:
        if value == "":
            return True
    return False

# Метод, що перевіряє чи починаються значення з заданого символа
def values_start_with(self, symbol, values):
    for value in values:
        if value[0] == symbol:
            return True
    return False

# Метод, що відображає діалогове вікно
def show_dialog(self, message):
    # Створення діалогового вікна
    self.dialog = MDDialog(
        text=message,  # текст діалогового вікна
        buttons=[
            MDFlatButton(  # кнопка діалогового вікна
                text="OK",
                theme_text_color="Custom",
                text_color="#1987ff",
                on_release=lambda _: self.dialog.dismiss()
                # закриття діалогового вікна при натисканні кнопки "OK"
            )
        ]
    )
```

```
        self.dialog.open()  # відкриття діалогового вікна

    # Метод, що повертає на попередній екран
    def navigate_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'MainScreen'

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value:  # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = True
        else:  # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

extensive_use_coefficient_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField


# Клас екрану для обчислення коефіцієнта екстенсивного використання
class ExtensiveUseCoefficientScreen(Screen):
    dialog = None  # змінна, що буде вказувати на діалогове вікно
    round_up_to_text_field = None  # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що розраховує коефіцієнт екстенсивного використання
    def calculate(self):
        # Отримання введених користувачем даних з полів для введення
        actual_worked_time_input = self.ids.actual_worked_time.text
        actual_fund_input = self.ids.actual_fund.text

        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([actual_worked_time_input, actual_fund_input]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
        elif self.values_start_with('-', [actual_worked_time_input, actual_fund_input]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
        elif self.values_start_with('0', [actual_worked_time_input, actual_fund_input]):
            self.show_dialog("Значення мають бути більше за нуль!")
        else:
            # Обчислення результату
            extensive_use_coefficient = int(actual_worked_time_input) / int(actual_fund_input)

            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                # Кількість знаків після коми, введена користувачем
                round_up_to_input = self.round_up_to_text_field.text
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
```

```
                self.show_dialog("Всі поля мають бути заповнені!")
            # Якщо значення починається з '-', то відобразити діалогове вікно
            elif round_up_to_input[0] == '-':
                self.show_dialog("Значення мають бути додатні!")
            # Якщо значення більше 7, то відобразити діалогове вікно
            elif int(round_up_to_input) > 7:
                self.show_dialog("Значення має бути не більше 7!")
            else:
                # Округлення результату
                extensive_use_coefficient = round(Decimal(str(extensive_use_coefficient)),
int(round_up_to_input))
        # Відображення результату
        self.ids.result.text = "Коефіцієнт екстенсивного використання: " + str(extensive_use_coefficient)


    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message,  # текст діалогового вікна
            buttons=[
                MDFlatButton(  # кнопка діалогового вікна
                    text="OK",
                    theme_text_color="Custom",
                    text_color="#1987ff",
                    on_release=lambda _: self.dialog.dismiss()
                    # закриття діалогового вікна при натисканні кнопки "OK"
                )
            ]
        )
        self.dialog.open()  # відкриття діалогового вікна

    # Метод, що повертає на попередній екран
    def navigate_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'MainScreen'

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value:  # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
```

```
            self.ids.round_layout.adaptive_height = True
        else:  # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

fund_availability_screen.py

```
from kivy.uix.screenmanager import Screen
from decimal import Decimal
from kivymd.uix.dialog import MDDialog
from kivymd.uix.button import MDFlatButton
from kivymd.uix.textfield import MDTextField


# Клас екрану для обчислення фондоозброєності
class FundAvailabilityScreen(Screen):
    dialog = None  # змінна, що буде вказувати на діалогове вікно
    round_up_to_text_field = None  # змінна, що буде вказувати на текстове поле для округлення

    # Метод, що розраховує фондоозброєність
    def calculate(self):
        # Отримання введених користувачем даних з полів для введення
        main_assets_average_annual_cost_input = self.ids.main_assets_average_annual_cost.text
        employees_input = self.ids.employees.text

        # Якщо хоча б одне поле порожнє, то відобразити діалогове вікно
        if self.values_are_empty([main_assets_average_annual_cost_input, employees_input]):
            self.show_dialog("Всі поля мають бути заповнені!")
        # Якщо хоча б одне значення починається з '-', то відобразити діалогове вікно
        elif self.values_start_with('-', [main_assets_average_annual_cost_input, employees_input]):
            self.show_dialog("Значення мають бути додатні!")
        # Якщо хоча б одне значення починається з '0', то відобразити діалогове вікно
        elif self.values_start_with('0', [main_assets_average_annual_cost_input, employees_input]):
            self.show_dialog("Значення мають бути більше за нуль!")
        # Якщо значення починається з '.', то відобразити діалогове вікно
        elif self.values_start_with('.', [main_assets_average_annual_cost_input]):
            self.show_dialog("Значення не може починатись з крапки!")
        else:
            # Обчислення результату
            fund_availability = float(main_assets_average_annual_cost_input) / int(employees_input)

            # Якщо опція округлення обрана
            if self.ids.round_switch.active:
                # Кількість знаків після коми, введена користувачем
                round_up_to_input = self.round_up_to_text_field.text
                # Якщо поле порожнє, то відобразити діалогове вікно
                if round_up_to_input == "":
                    self.show_dialog("Всі поля мають бути заповнені!")
                # Якщо значення починається з '-', то відобразити діалогове вікно
                elif round_up_to_input[0] == '-':
                    self.show_dialog("Значення мають бути додатні!")
                # Якщо значення більше 7, то відобразити діалогове вікно
                elif int(round_up_to_input) > 7:
                    self.show_dialog("Значення має бути не більше 7!")
                else:
                    # Округлення результату
                    fund_availability = round(Decimal(str(fund_availability)), int(round_up_to_input))
            # Відображення результату
            self.ids.result.text = "Фондоозброєність: " + str(fund_availability)

    # Метод, що перевіряє чи є рядки порожніми
    def values_are_empty(self, values):
        for value in values:
```

```
            if value == "":
                return True
        return False

    # Метод, що перевіряє чи починаються значення з заданого символа
    def values_start_with(self, symbol, values):
        for value in values:
            if value[0] == symbol:
                return True
        return False

    # Метод, що відображає діалогове вікно
    def show_dialog(self, message):
        # Створення діалогового вікна
        self.dialog = MDDialog(
            text=message,  # текст діалогового вікна
            buttons=[
                MDFlatButton(  # кнопка діалогового вікна
                    text="OK",
                    theme_text_color="Custom",
                    text_color="#1987ff",
                    on_release=lambda _: self.dialog.dismiss()
                    # закриття діалогового вікна при натисканні кнопки "OK"
                )
            ]
        )
        self.dialog.open()  # відкриття діалогового вікна

    # Метод, що повертає на попередній екран
    def navigate_back(self):
        self.manager.transition.direction = 'right'
        self.manager.current = 'MainScreen'

    # Метод, що викликається при зміні стану перемикача для округлення
    def on_round_switch_active(self, checkbox, value):
        if value:  # якщо перемикач ввімкнений
            # Створення текстового поля для округлення
            self.round_up_to_text_field = MDTextField(
                hint_text="Кількість знаків після коми",
                input_filter="int",
                helper_text="(від 0 до 7)",
                helper_text_mode="persistent"
            )
            # Відображення текстового поля для округлення
            self.ids.round_layout.add_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = True
        else:  # інакше
            # Прибирання текстового поля для округлення з екрану
            self.ids.round_layout.remove_widget(self.round_up_to_text_field)
            self.ids.round_layout.adaptive_height = False
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
ПЗ_Веселов.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
ПЗ_Веселов.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
calculating_economic_indicators_of_enterprise_app.zip	Архів. Містить вихідний код програмного забезпечення
calculating_economic_indicators_of_enterprise_app-0.1-arm64-v8a_armeabi-v7a-debug.apk	Файл в форматі APK для встановлення ПЗ на операційну систему Android
main.exe	Файл в форматі .exe для запуску ПЗ на операційній системі Windows
Презентація	
Презентація_Веселов.pptx	Презентація кваліфікаційної роботи