Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

РОЗРОБКА КРОСПЛАТФОРМЕННОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ PYTHON

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувача другого (магістерського)
рівня вищої освіти
Гончарука Максима Олеговича

НАУКОВИЙ КЕРІВНИК:
кандидат педагогічних наук, доцент
Габрусєв В. Ю

РЕЦЕНЗЕНТ:
доцент кафедри вищої математики
ТНТУ імені Івана Пулюя, кандидат
технічних наук, доцент
Ірина ГАБРУСЄВА

Тернопіль – 2025

АНОТАЦІЯ

Гончарук М. О. Розробка кросплатформенного додатку для вивчення мови програмування Python. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 122 Комп'ютерні науки. ТНПУ ім. В. Гнатюка. Тернопіль, 2025. 58 с.

У кваліфікаційній роботі розглянуто процес розробки кросплатформенного програмного додатку для вивчення мови програмування Python. Проаналізовано актуальність використання Python як однієї з провідних мов програмування для навчання, а також переваги кросплатформенних рішень у сучасних програмних продуктах. Обґрунтовано вибір ігрового рушія Godot як основного інструменту розробки та досліджено можливості його інтеграції з серверним застосунком Firebase.

Розроблено структуру додатку з адаптивним дизайном для різних платформ, реалізовано взаємодію з серверною частиною для збереження та обробки даних користувачів, а також здійснено вбудування інтерпретатора Python для створення початкового інтерактивного навчального середовища. Отримані результати підтверджують доцільність використання кросплатформенного підходу для підвищення доступності та ефективності навчання програмуванню.

Ключові слова: кросплатформенний додаток, Python, Godot, Firebase, навчальні системи, програмування.

ABSTRACT

Honcharuk M. O. Development of a cross-platform application for learning the Python programming language. Qualification work for obtaining a master's degree in the specialty 122 Computer Science. Ternopil Volodymyr Hnatiuk National Pedagogical University. Ternopil, 2025. 58 p.

The qualification work focuses on the development of a cross-platform software application for learning the Python programming language. The relevance of Python as one of the leading programming languages for education is analyzed, along with the advantages of cross-platform solutions in modern software products. The choice of the Godot engine as the main development tool is substantiated, and its integration capabilities with the Firebase backend service are explored.

The application structure with an adaptive design for multiple platforms was developed, server-side interaction for storing and processing user data was implemented, and a built-in Python interpreter was integrated to create an initial interactive learning environment. The obtained results confirm the effectiveness of the cross-platform approach in increasing accessibility and improving the efficiency of programming education.

Keywords: cross-platform application, Python, Godot, Firebase, educational systems, programming.

ЗМІСТ

ВСТУП

Стрімкий розвиток цифрових технологій та зростання потреби у фахівцях з інформаційних технологій актуалізують удосконалення підходів до навчання програмування в закладах освіти та в системі неформального навчання. Особливу увагу зосереджено на мові програмування Python, що завдяки простому синтаксису, широкому спектру бібліотек і універсальності застосування використовується в аналізі даних, машинному навчанні, веброзробці, автоматизації процесів та створенні прикладних програмних рішень. За результатами міжнародних опитувань спільноти розробників Python стабільно входить до переліку найпопулярніших мов програмування у світі станом на 2024–2025 роки.

Дослідження у сфері розробки програмних застосунків мовою Python представлені працями таких зарубіжних науковців і практиків як Р. Базюк, П. Баррі, О. Башуцька, Д. Бізлі, Р. Геттінґер, С. Довгопол, Л. Івашкевич, М. Лутц, О. Перепелиця, О. Писатков, Л. Рамальйо, Ґ. ван Россум, А. Свейґарт, Д. Трейтяк, І. Товсточуб, С. Шокалюк, Ю. Хлапонін. Напрям кросплатформенної розробки, створення навчальних застосунків і цифрових ігор у середовищі Godot досліджують К. Бредфілд., В. Коцовський, А. Торн. Праці Д. Маркес, М. Тркулья, охоплюють питання використання рушія Godot, мови сценаріїв GDScript та практичної побудови 2D і 3D кросплатформенних проєктів. Розвиток ігрових рушіїв та інтерактивних платформ загального призначення представлений працями А. Манзура та Д. Маркеса як ключових учасників розроблення Godot і авторів навчально-методичних матеріалів.

Python вирізняється простим синтаксисом, потужними функціональними можливостями та широким спектром застосування: від веброзробки до наукових досліджень і штучного інтелекту. Саме ці якості роблять її ідеальним вибором для першого знайомства зі світом

програмування. А в умовах мобільності здобувачів освіти та різноманітності технічних пристроїв актуальною стає розробка кросплатформенних програмних продуктів, що забезпечують можливість навчання незалежно від операційної системи чи типу пристрою. Подібний підхід дозволяє оптимізувати витрати на розробку програмного забезпечення, забезпечити єдність навчального контенту та покращити доступність освітніх ресурсів.

Інтерактивні навчальні додатки створюють умови для поєднання теоретичного матеріалу з практичним виконанням завдань, миттєвим отриманням зворотного зв’язку та індивідуалізацією траєкторії навчання користувачів. Попри наявність значної кількості онлайн-курсів, відеоплатформ та навчальних середовищ, спостерігається дефіцит якісних автономних кросплатформенних додатків українською мовою, які поєднують структурований курс вивчення Python, інтерактивні вправи та вбудоване середовище виконання коду. Зазначена суперечність зумовлює необхідність розроблення власного програмного рішення педагогічного спрямування.

У рамках кваліфікаційної роботи розроблено інноваційне програмне рішення – кросплатформений додаток для вивчення Python, який поєднує елементи інтерактивного середовища та гейміфікації. Особливістю запропонованого підходу є інтеграція з хмарними сервісами для збереження результатів навчання, а також реалізація вбудованого інтерпретатора Python, що дозволяє виконувати код безпосередньо у додатку.

Метою дослідження є розробка та дослідження ефективності кросплатформеного додатку для вивчення мови програмування Python, який забезпечує зручне, доступне та індивідуалізоване навчальне середовище.

Завдання дослідження:

1. Проаналізувати переваги використання кросплатформених додатків у процесі навчання програмуванню;

2. Визначити актуальність та переваги вивчення Python як базової мови програмування;

3. Дослідити інструменти для створення кросплатформених додатків;

4. Розробити адаптивний інтерфейс та інтерактивні навчальні модулі;

5. Реалізувати вбудований інтерпретатор Python;

6. Перевірити ефективність запропонованого додатку шляхом тестування з реальними користувачами.

Об'єкт дослідження є процес навчання мовам програмування в умовах цифрової трансформації освіти.

Предмет дослідження є технології створення кросплатформених додатків для вивчення мови програмування Python та їх вплив на ефективність навчального процесу.

Результати цієї роботи можуть бути корисними для широкого кола користувачів: викладачів, здобувачів, школярів, які вивчають програмування, а також розробників освітніх платформ, методистів та освітніх менеджерів, зацікавлених у впровадженні інноваційних інструментів у навчальний процес.

Кваліфікаційна робота складається з змісту, вступу, двох розділів з висновками, загальних висновків і списку використаних джерел, та налічує 60 сторінок.

РОЗДІЛ 1
PYTHON ЯК ПРЕСТИЖНА МОВА ПРОГРАМУВАННЯ ДЛЯ НАВЧАННЯ

1.1 Переваги кросплатформенності у додатках

У сфері інформаційних технологій спостерігається зростання значущості кросплатформеної розробки програмних продуктів. Такий підхід передбачає створення програмного забезпечення, здатного стабільно працювати на різних операційних системах на основі єдиної програмної логіки. Особливого значення набуває в освітній галузі, оскільки універсальний доступ до цифрових ресурсів і навчальних інструментів визначає ефективність освітнього процесу та забезпечує рівні умови для всіх учасників навчання.

Термін «кросплатформенність» (англ. Cross-platform) походить від поєднання слів cross – «перехресний» або «той, що охоплює кілька» – і platform – «платформа», тобто операційна система або середовище, на якому виконується програмне забезпечення. Раннє використання поняття пов’язане з розробкою програмного забезпечення для різних комп’ютерних платформ, що ставало актуальним у 1980-1990-х роках, коли різноманіття операційних систем збільшувалося, і з’явилася потреба в розробці універсальних рішень.

Зародження терміну «кросплатформенність» з’явилося разом з потребою вирішення проблеми сумісності програм на різних платформах, і процес був частиною еволюції програмних продуктів, зокрема для бізнесу і консалтингових компаній, які прагнули знижувати витрати на розробку програмного забезпечення для різних операційних систем. Відповідно, розробникам доводилося створювати окремі версії програм для кожної популярної операційної системи, саме тоді й виникла ідея створення

програм, які могли б працювати на різних системах, що й стало основою поняття «кросплатформенності».

Проаналізувавши обрану тематику, можемо виокремити, що у технічній документації – термін широко використовується в офіційній документації до фреймворків і мов програмування, де наголошується на можливості розробки кросплатформенних рішень. У наукових працях та технічній літературі – у працях, присвячених розробці програмного забезпечення, інженерії ПЗ, мобільній та веброзробці, термін використовується для класифікації підходів до створення програм (нативна, гібридна, кросплатформенна розробка).

Наприклад, В. Коцовський дає визначення кросплатформненності – це «властивість програмного забезпечення працювати більш ніж на одній програмній (в тому числі – операційній системі) або апаратній платформи, та технології, що дозволяють досягти цієї властивості» [7, с. 1].

У навчальних курсах та стандартах освіти – термін включається до курсів з програмування, розробки програмного забезпечення, інженерії програмних систем, особливо в контексті вивчення сучасних технологій розробки додатків. Також згадується у галузевих медіа та IT-ресурсах – такі ресурси, як TechCrunch, Stack Overflow, GitHub, Medium (розділ програмування), часто публікують статті про переваги, порівняння фреймворків, тренди у кросплатформенній розробці.

Виокремимо, що фреймворк (framework) – це набір інструментів, бібліотек та правил, який використовується для створення програмних додатків. Зазвичай являє собою структуру, яка визначає, як компоненти програми повинні взаємодіяти між собою, які шаблони використовувати для створення інтерфейсів і які методи методи використовувати для роботи з базами даних та іншими зовнішніми ресурсами.

Отже, кросплатформний застосунок – це програмне забезпечення, яке може працювати на кількох різних платформах і операційних системах, таких як Windows, Mac OS, Linux, iOS та Android. Основна ідея

кросплатформенності полягає у створенні універсального програмного забезпечення, яке легко підтримувати.

Загалом, у додатках кросплатформенність – це стратегічна перевага, яка дозволяє пришвидшити вихід продукту на ринок, зменшити витрати та забезпечити ширше охоплення аудиторії без шкоди для якості та функціональності.

Перші серйозні спроби зробити програмне забезпечення кросплатформеним були пов’язані з мовами програмування, які мали на меті забезпечити сумісність між різними операційними системами. Однією з таких мов була C, яка на початку 1980-х років стала основною для багатьох операційних систем, таких як UNIX, MS-DOS і Macintosh. Мова C дозволяла писати програми, які могли компілюватися і виконуватися на різних платформах з мінімальними змінами в коді.

У 1980-х роках компанії почали розробляти програмні інтерфейси, що дозволяли забезпечити спільну основу для програм, яка могла адаптуватися під різні ОС. Одним із прикладів таких технологій була X Window System, система для створення графічних інтерфейсів користувача на UNIX – платформах, що дозволяла працювати з різними апаратними платформами.

Ідея створення віртуальних машин дозволила досягти кросплатформенності ще одним способом. Найвідоміший приклад – Java, яка була представлена у 1995 році компанією Sun Microsystems. Java дозволяла писати програмний код, який компілювався в байт-код, що може виконуватися на будь-якій платформі, де є інтерпретатор або віртуальна машина Java (JVM), що дозволило програмам працювати незалежно від операційної системи.

У середині 1990-х і 2000-х років з’явилися різні фреймворки та середовища, які спрощували розробку кросплатформенних програм, наприклад, Qt (фреймворк для розробки графічних інтерфейсів) і Electron (фреймворк для створення кросплатформенних настільних додатків на базі

веб-технологій). У 2010-х роках з’явилися мобільні фреймворки, які дозволяли створювати кросплатформенні додатки для Android та iOS.

Ще одним важливим напрямом у досягненні кросплатформенності стали веб-технології, які дозволяють створювати додатки, що працюють у браузері, незалежно від операційної системи. HTML, CSS і JavaScript стали основними інструментами для розробки кросплатформенних веб-додатків, які можуть працювати на будь-якому пристрої з веб-браузером.

Розробка міжплатформних додатків зробила революцію у створенні додатків, дозволивши компаніям скоротити витрати на розробку в середньому на 42% і скоротити час виходу на ринок приблизно на 35%. Сучасні фреймворки, такі як Flutter, еволюціонували, щоб забезпечити продуктивність, порівнянну з нативними програмами, що дозволяє таким компаніям, як Instagram і Uber Eats, ефективно використовувати свої можливості. У міру зрілості ландшафту організації можуть приймати обґрунтовані рішення щодо свого технологічного стеку, зосереджуючись на основних функціях і масштабованості відповідно до своїх бізнес-цілей [17].

Зазначаємо, що розробка кросплатформенних додатків дозволяє значно скоротити витрати часу та фінансів, оскільки одна команда розробників може створити єдиний код, який працюватиме на різних платформах, що усуває потребу в окремих командах для кожної операційної системи, що знижує витрати на розробку та обслуговування додатку .

У праці «The Six Most Popular Cross-Platform App Development Frameworks» [47] висвітлено шість провідних кросплатформних фреймворків, включаючи:

- Flutter (від Google використовує Dart);
- Ionic (використовує мову програмування JavaScript);
- Kotlin Multiplatform;
- NativeScript;
- NET MAUI (від Microsoft (раніше Xamarin) використовує C# і XAML);

- React Native (використовує JavaScript як мову програмування).

Кожен з яких має унікальні функції та переваги, як результат приклади популярних кросплатформних додатків включають:

- Flutter - Google Ads, My BMW App, eBay Motors, New York Times;
- React Native - Instagram, Skype, Walmart, Airbnb, Discord, Tesla, Shopify;
- Xamarin - Світовий банк, Fox Sports, Alaska Airlines, BBC Good Food [33].

Обираючи фреймворк, розробники повинні враховувати такі фактори, як сумісність мов програмування, довгострокова підтримка, можливості налаштування інтерфейсу користувача, безпека та доступність навчальних ресурсів, щоб переконатися, що вибране рішення відповідає вимогам їх проекту.

«Метою кросплатформної розробки додатків є націлювання на різні операційні системи в одному проекті. Ви створюєте ці додатки за допомогою міжплатформних фреймворків, які використовують SDK для певної платформи (SDK для Android та iOS SDK) з єдиного API, що дає вам змогу легко отримати доступ до SDK і бібліотек різних платформ» [33].

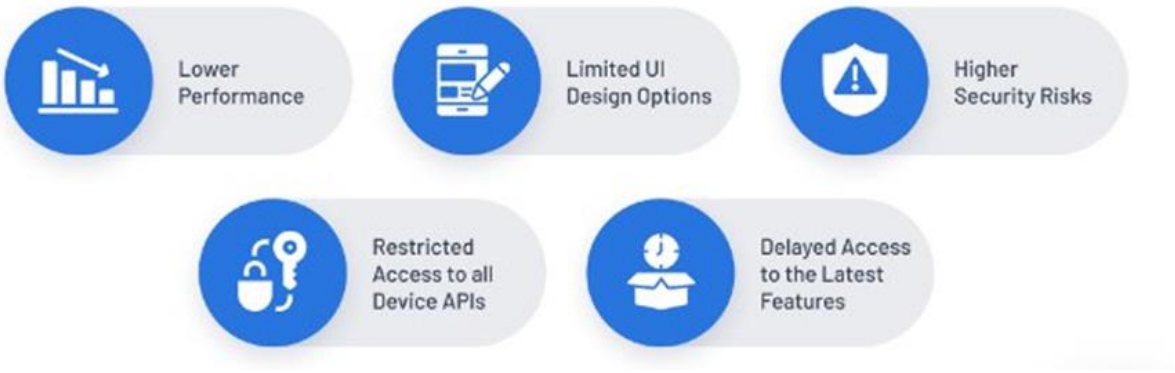

Рис. 1.1 Переваги кросплатформенності

Завдяки кросплатформенності, додаток стає доступним для користувачів різних пристроїв та операційних систем, що дозволяє залучити ширшу аудиторію, що особливо важливо для освітніх додатків, які мають

бути доступними для здобувачів та викладачів незалежно від використовуваних ними пристроїв.

Наприклад розробка додатків, які безперебійно працюють на кількох платформах, має вирішальне значення в сучасному стрімкому цифровому світі, де користувачі постійно отримують доступ до вмісту на різноманітних пристроях, використовуючи низку операційних систем. Згідно з даними StatCounter «Android наразі займає більшу частку в 71,09%, за нею йде iOS (27,68%), Samsung (0,25%), Linux і KaiOS становлять менший відсоток (0,01 і 0,02 відсотка)» [32].

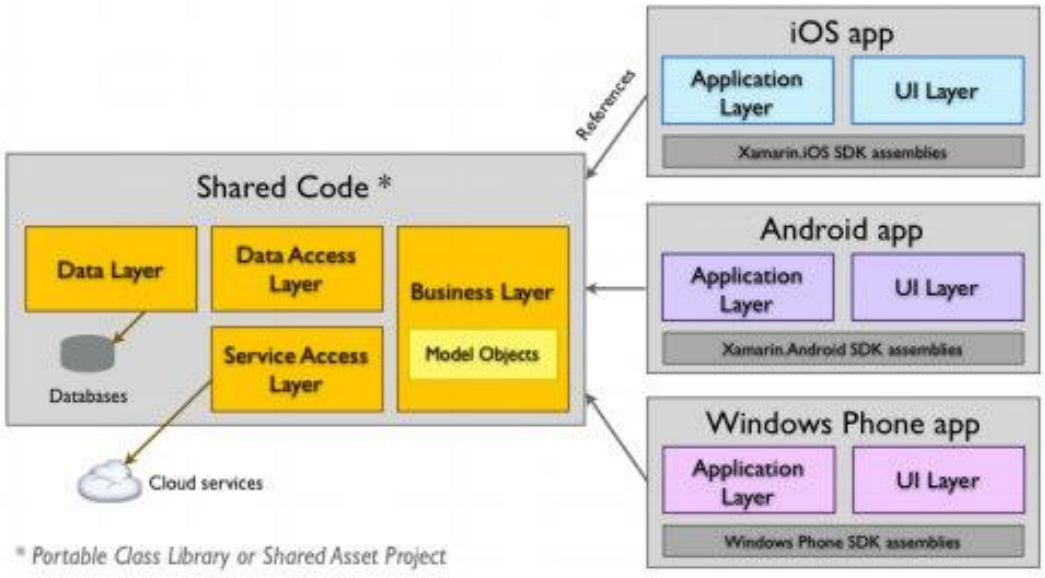

Рис. 1.2 Структура кросплатформенного підключення

Зважаючи на таке розмаїття пристроїв і операційних систем, розробникам доводиться шукати ефективні підходи для оптимізації процесу створення програмного забезпечення. Особливо корисно для стартапів і невеликих команд, які прагнуть швидко вийти на ринок із якісним продуктом, охоплюючи широку аудиторію. Кросплатформенна розробка забезпечує єдиний користувацький інтерфейс та узгоджений досвід взаємодії на всіх платформах, що підвищує довіру та задоволеність користувачів, оскільки вони отримують однаковий функціонал та дизайн незалежно від пристрою .

Рис. 1.3. 6 ключових переваг реалізації кросплатформенного додатку

Однією з ключових переваг комп'ютерних додатків є зменшення витрат на розробку та підтримку програмного забезпечення. Замість створення окремих версій додатку для кожної операційної системи, розробники працюють з однією кодовою базою, що спрощує оновлення, виправлення помилок та впровадження нових функцій та значно підвищує ефективність команди розробників. Оновлення можуть бути швидко розгорнуті на всіх платформах одночасно, що забезпечує актуальність та стабільність додатку .

Крім того, кросплатформенні комп'ютерні додатки забезпечують однаковий користувацький досвід на всіх системах, що важливо для навчальних, професійних або бізнес-застосунків, де узгодженість інтерфейсу і функціональності забезпечує зручність для користувачів, незалежно від того, на якому пристрої вони працюють.

«Кросплатформеними можна назвати більшість сучасних високорівневих мов програмування. Наприклад, C, C ++, Java – кросплатформені мови на рівні компіляції, тобто для цих мов є компілятори під різні платформи, що дозволяє, при належній якості коду, не переписувати основний двигун програми, змінюються лише особливі системозалежні частини» [8].

Ще одна перевага – простота розгортання. Багато сучасних інструментів, таких як Electron, Qt або .NET MAUI, дозволяють упакувати

додаток для різних платформ з мінімальними змінами в коді, а отже це скорочує час виходу продукту на ринок і полегшує розповсюдження.

Уніфікований кодовий базис та спрощене обслуговування призводять до зниження витрат на технічну підтримку. Менша кількість потенційних помилок та узгодженість функціоналу на різних платформах зменшують потребу в частих зверненнях до служби підтримки [46]. «Не менш важливі для платформ є стандартизовані бібліотеки середовища виконання. Зокрема, стандартом стала бібліотека мови Сі. З великих кроссплатформенних бібліотек - Qt, GTK +, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL» [8].

Кросплатформенна розробка дозволяє використовувати єдиний стек технологій для всіх платформ, що спрощує процес навчання та адаптації розробників. Зазначимо, що це особливо корисно в освітньому контексті, де здобувачі можуть зосередитися на вивченні однієї технології, застосовуючи її для створення додатків на різних платформах.

Стек технологій – це набір інструментів, технологій і програмного забезпечення, які використовуються для розробки та запуску програм. У контексті кросплатформенної розробки стек технологій дозволяє використовувати один набір інструментів для створення додатків.

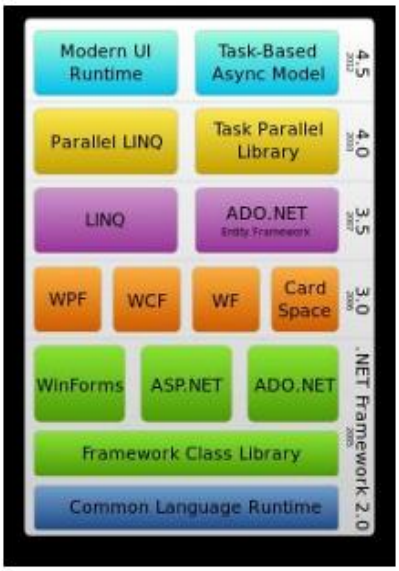

Рис. 1.4 Оновлення стек технологій

Стек технологій може включати різні компоненти, в залежності від типу додатка та вимог до нього. Зазвичай він складається з декількох основних рівнів, таких як:

1. Мова програмування – це основний інструмент для написання коду. У кросплатформенній розробці часто використовуються мови, які підтримують виконання на різних операційних системах, а саме:

- C# – часто використовується в кросплатформенних фреймворках, таких як Xamarin;
- C++ та Java також застосовуються в деяких кросплатформенних розробках, наприклад, в ігрових двигунах або великих програмних системах;
- Dart – мова програмування, що використовується в Flutter для кросплатформенної розробки мобільних додатків;
- JavaScript – часто використовується у фреймворках для мобільних додатків (наприклад, React Native) або веб-додатків.

2. *Фреймворки та бібліотеки* – це готові набори інструментів, що значно спрощують розробку додатків, за рахунок надання попередньо реалізованих рішень для часто використовуваних функцій і задач. У кросплатформенній розробці використовуються такі популярні фреймворки:

- Electron – для створення настільних додатків за допомогою веб-технологій (HTML, CSS, JavaScript);
- Flutter – для створення кросплатформенних мобільних додатків з використанням мови Dart;
- Qt – кросплатформенний фреймворк для розробки додатків для настільних систем;
- React Native – для розробки мобільних додатків з використанням JavaScript і React;
- Xamarin – дозволяє писати додатки на C# і запускати їх на iOS, Android і Windows.

3. *Інструменти для побудови та тестування*:

- Fastlane – для автоматизації процесу збірки і публікації мобільних додатків;
- Gradle та Maven для управління залежностями і автоматизації процесу побудови в Java;
- JUnit, Mockito, Jest, Appium – для тестування коду в Java, JavaScript і інших мовах;
- Webpack для упаковки ресурсів у веб-додатках.

4. *Інтерфейси для графічних користувачів* (UI/UX). Створення кросплатформенних додатків передбачає, що інтерфейс користувача (UI) буде однаковим на всіх платформах. Багато фреймворків і бібліотек надають вбудовані компоненти для створення інтерфейсу:

- Flutter і React Native мають власні набори віджетів для створення UI, який буде виглядати нативно на кожній платформі;
- Qt має набір віджетів для створення графічних інтерфейсів, які працюють однаково на різних платформах.

5. *Бази даних і сервери*. Якщо додаток має працювати з даними або потребує серверної частини, вибір бази даних та серверної технології також є частиною стека:

- GraphQL або RESTful API для взаємодії між клієнтською та серверною частиною;
- Node.js, Django, Flask, Spring – серверні фреймворки, які дозволяють створювати серверну частину додатка;
- SQL або NoSQL бази даних (наприклад, MySQL, PostgreSQL, MongoDB).

6. *Інструменти для DevOps і хмарних обчислень*. Інструменти для автоматизації розгортання і управління інфраструктурою також є важливою частиною сучасного стека:

- AWS, Google Cloud, Azure – хмарні сервіси для хостингу додатків, баз даних;

- Docker – для створення контейнеризованих додатків, які можуть запускатися на різних платформах;
- Kubernetes – для оркестрації контейнерів в масштабних системах.

У результаті, кросплатформенність стає стратегічною перевагою для стартапів і компаній, які прагнуть до швидкого масштабування або запуску MVP-продукту, водночас забезпечуючи ширше охоплення ринку та гнучкість у адаптації до змін у технологічному середовищі. Загалом, кросплатформенність у комп'ютерних додатках – це не лише про зручність для розробника, а й про гнучкість, що дозволяє створювати доступні, стабільні та масштабовані рішення для максимальної кількості користувачів..

Кросплатформенні додатки – це програмні продукти, які можуть працювати на кількох операційних системах або платформах без необхідності змінювати вихідний код для кожної з них. Такі додатки розробляються з використанням спеціальних фреймворків або технологій, що дозволяють створити єдину кодову базу, яку можна запускати на різних пристроях – комп'ютерах, смартфонах, планшетах тощо. Наприклад, один і той самий кросплатформенний додаток може працювати як на Windows, так і на macOS або Linux.

Проведений аналіз дозволяє зробити висновок, що кросплатформенність виступає однією з провідних тенденцій сучасної розробки програмного забезпечення, зокрема в освітній галузі. Використання єдиної кодової бази для створення додатків, здатних функціонувати на різних операційних системах і пристроях, забезпечує суттєве скорочення часових та фінансових витрат на розробку й супровід програмних продуктів, спрощує процес оновлення функціоналу та підвищує стабільність роботи застосунків.

Універсальність кросплатформенних рішень сприяє розширенню кола користувачів освітніх додатків незалежно від типу використовуваного обладнання, що особливо важливо в умовах дистанційного та змішаного навчання. Забезпечення єдиного інтерфейсу й узгодженого користувацького

досвіду підвищує доступність навчальних ресурсів, комфортність роботи з ними та загальну результативність освітнього процесу.

Отримані результати засвідчують, що застосування сучасних фреймворків кросплатформенної розробки створює умови для швидкого впровадження інноваційних освітніх рішень, орієнтованих на інтерактивність, масштабованість та гнучкість використання.

1.2 Актуальність вивчення Python з використанням додатків

У контексті динамічного розвитку інформаційних технологій програмування розглядається не лише як інструмент фахової діяльності, а й як складова цифрової грамотності, що визначає готовність особистості до функціонування в умовах цифровізованого суспільства. Однією з найпопулярніших мов програмування є Python – мова, що відзначається простою синтаксичною структурою, універсальністю та широким спектром застосування. У зв’язку з цим зростає інтерес до вивчення Python серед різних категорій населення: від школярів і здобувачів до фахівців з інших галузей. Особливу роль у цьому процесі відіграють сучасні додатки, що полегшують та урізноманітнюють процес навчання.

Python активно застосовується в таких сферах, як веброзробка, аналіз даних, штучний інтелект, автоматизація процесів та створення додатків для навчання, так як зрозуміла структура робить мову доступною як для початківців, так і для досвідчених розробників.

«Мову програмування Python створив нідерландський програміст Г. Россум (Guido van Rossum) у грудні 1989 року. Він розпочав розробку Python як хобі-проєкт під час різдвяних канікул, прагнучи створити мову, яка була б наступником мови ABC, але з кращою підтримкою для системних викликів і взаємодії з операційною системою Unix. Назву «Python» він обрав на честь британського комедійного шоу «Monty Python’s Flying Circus» («Летючий цирк М. Пайтона») [14].

Джерело E-lab.com зазначає, що «першу публічну версію Python (0.9.0) Г. Россум оприлюднив у лютому 1991 року, працюючи в Центрі математики та інформатики (Centrum Wiskunde & Informatica, CWI) в Нідерландах. Дана версія вже містила основні функціональні можливості, такі як класи, винятки, функції та основні типи даних, що стали фундаментом для подальшого розвитку мови» [6].

Python посідає провідні позиції серед мов програмування, що застосовуються для навчання, завдяки структурованості, зрозумілому синтаксису та високій читабельності програмного коду. Такі властивості забезпечують сприятливі умови для опанування програмування на початковому етапі, оскільки здобувачі можуть концентрувати увагу на засвоєнні алгоритмічного мислення та принципів побудови програм, не відволікаючись на опрацювання складних мовних конструкцій.

Крім того, Python підтримується великою спільнотою, має багатий набір бібліотек та фреймворків, що охоплюють різноманітні сфери застосування: від веброзробки до машинного навчання й аналізу даних, що дозволяє здобувачам на практиці застосовувати знання, здобуті в процесі навчання, та розвивати свої проєкти, орієнтуючись на реальні потреби ринку.

Python активно використовується в провідних університетах світу як базова мова для вивчення програмування, алгоритмів і структур даних. Така популярність підтверджує її актуальність і освітню цінність, адже вона дає змогу ефективно формувати фундаментальні навички програмування.

Окрім освітньої привабливості, Python має високий професійний потенціал. Багато компаній, включаючи світових лідерів у сфері технологій, використовують цю мову у своїх продуктах і сервісах, що відкриває широкі кар’єрні перспективи для тих, хто опановує Python на етапі навчання.

Рис. 1.5 Why Python

«Python – це високорівнева, інтерпретована мова програмування, яка підтримує кілька парадигм програмування» [14], та має численні переваги:

- активна спільнота користувачів;
- величезна кількість бібліотек і фреймворків (наприклад, Django, Flask та Web2Py.) для різноманітних сфер (штучний інтелект, веб-розробка, автоматизація, наука про дані).
- гнучкість, так як з її допомогою можна обробляти дані, створювати моделі, робити вебсайти тощо;
- ефективність розробки;
- можна побудувати бізнес-логіку та взаємодію з базою даних на використовувати для роботи з графікою;
- попит на програмістів Python на ринку праці;
- простота у вивченні завдяки читабельному синтаксису.

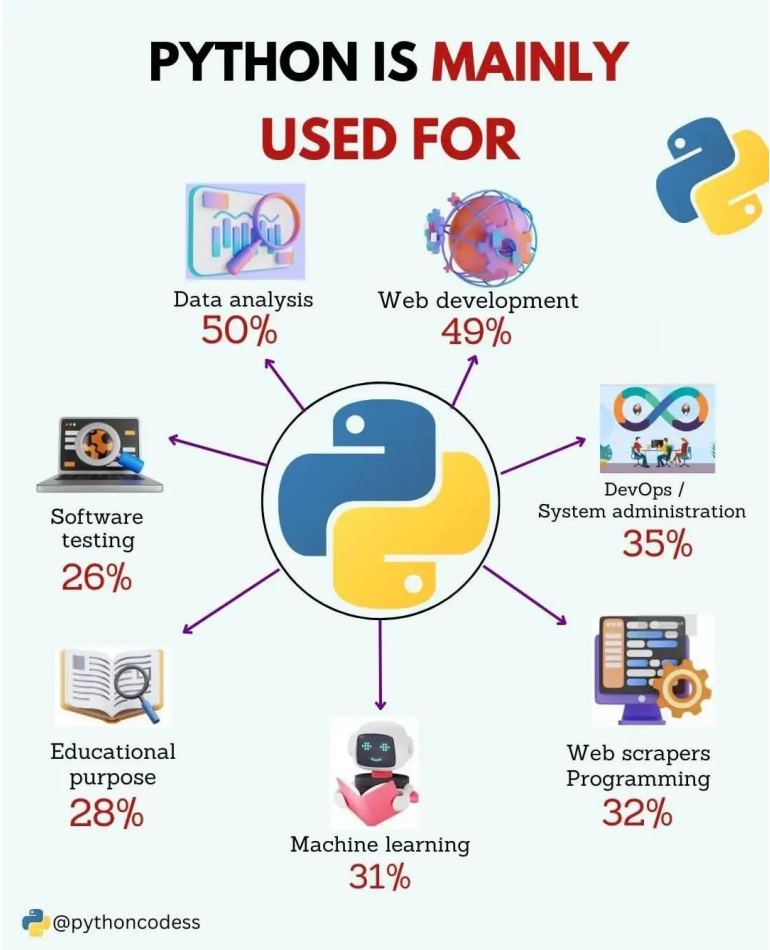

Рис.1.6 Основні сфери використання Python

За даними опитування Stack Overflow «у 2023 році JavaScript одинадцятий рік поспіль продовжує позицію найпоширенішої мови програмування. Python витіснив SQL як третю найпоширенішу мову, але посідає перше місце для тих, хто не є професійним розробником або не вчиться кодувати (інші кодери)» [43], що підтверджує практичну значущість. На Python створено багато сервісів, як-от Dropbox, Facebook, Instagram, Spotify, YouTube та Reddit та інші.

Освітні технології створили нові можливості для вивчення програмування. Навчальні додатки забезпечують інтерактивність, гейміфікацію та гнучкість, що дозволяє:

- брати участь у онлайн-курсах і інтерактивних уроках (наприклад, на DataCamp, SoloLearn, Coursera, edX);

- вивчати Python у зручному темпі, обираючи складність і час навчання індивідуально;
- відстежувати прогрес і мотивацію через систему балів, досягнень і нагород;
- долучатися до гейміфікованого навчання (наприклад, CodeCombat або CheckiO, де код використовується для проходження рівнів або розв’язання ігор);
- застосовувати знання на практиці у форматі вправ, задач та проєктів (наприклад, Codewars, HackerRank, Exercism);
- отримувати миттєвий зворотний зв’язок завдяки автоматичній перевірці коду;
- отримувати сертифікати про проходження курсів, які визнаються роботодавцями;
- практикувати написання коду без необхідності встановлення складного ПЗ, усе працює прямо в браузері або мобільному додатку (наприклад, Replit чи PythonAnywhere).;
- спілкуватися з однодумцями у спільнотах та форумах, де можна ставити питання й отримувати підтримку (наприклад, Stack Overflow, Reddit, Discord-спільноти для Python-розробників);
- створювати власні проєкти у візуальному середовищі, як-от Thonny або Trinket, де можна одразу бачити результат.

Узагальнюючи наведене, можна констатувати, що розвиток цифрових освітніх технологій суттєво трансформував підходи до опанування програмування. Навчальні програмні платформи забезпечують адаптивність, доступність і варіативність форм взаємодії, поєднуючи теоретичні компоненти з практичними завданнями.

Інтерактивні середовища, вбудовані інструменти автоматичної перевірки та гейміфіковані механізми сприяють формуванню стійкої мотивації до навчання та створюють умови для поступового, самостійно

регульованого опанування Python. Такі можливості сприяють розвитку алгоритмічного мислення, формуванню навичок практичного застосування знань і поступовому входженню здобувачів у спільноту програмістів, що, у свою чергу, підвищує ефективність навчального процесу та сприяє підготовці компетентних фахівців у сфері цифрових технологій.

Цифровізація суспільства активно інтегрує інноваційні програмні засоби для формування практичних навичок програмування, щоб полегшити процес навчання Python, створено низку корисних застосунків, доступних як онлайн, так і у вигляді мобільних додатків. Приклади доступних додатків для вивчення Python:

- Codecademy - пропонує інтерактивний курс Python, який веде здобувачів через основи мови з можливістю практикуватися у браузері;
- Coursera – пропонує курси Python від провідних університетів та коледжів, що покривають різні рівні від початківця до просунутого;
- Jupyter Notebook [37] - потужне середовище для вивчення Python у контексті аналізу даних і наукових досліджень;
- Mimo [31] – платформа, орієнтована на короткі інтерактивні уроки з миттєвими вправами;
- Replit [41] – онлайн-редактор з підтримкою Python, що дозволяє писати, запускати та ділитися кодом без встановлення IDE;
- SoloLearn – мобільний додаток, що пропонує курси з Python, інтерактивні вправи, спільноту для обговорень;
- Udemy – має велику кількість курсів на будь-який смак та рівень знань - від основ програмування до спеціалізованих тем, як-от машинне навчання, веброзроблення, автоматизація і багато іншого.

Варто зазначити, що цифрова трансформація освітнього середовища сприяла широкому впровадженню інноваційних інструментів, орієнтованих на підтримку процесу оволодіння програмуванням. Розроблені програмні платформи та навчальні додатки забезпечують доступність, мобільність і

різнорівневу підтримку користувачів, пропонуючи можливості для самостійного або структурованого навчання.

Завдяки інтерактивним завданням, гнучким навчальним траєкторіям, вбудованим системам зворотного зв’язку та спільнотам практиків формуються умови для поглибленого засвоєння Python та розвитку практичних компетентностей. Такий підхід не лише підсилює мотивацію до навчання, а й сприяє формуванню професійної готовності до подальшої діяльності у сфері програмування, що відповідає запитам сучасного цифровізованого суспільства.

Інтеграція цифрових технологій у процес підготовки фахівців з інформаційних технологій потребує переосмислення традиційних підходів до навчання програмування. Переваги використання додатків у навчальному процесі є:

- використання анімацій, схем, симуляцій сприяє кращому розумінню абстрактних концепцій;
- використання ігрових елементів або гейміфікація (досягнення, бали, рівні) стимулює мотивацію до навчання та формує сталі навчальні звички;
- доступ до великої кількості навчальних ресурсів – інтерактивних уроків, відеоматеріалів, віртуальних лабораторій, форумів і спільнот, що значно розширює навчальне середовище;
- індивідуалізація навчання, додатки дозволяють адаптувати освітній контент до рівня підготовки, темпу засвоєння знань та індивідуальних потреб здобувача освіти;
- можливість використання на різних платформах (мобільні пристрої, планшети, комп’ютери);
- підвищення зацікавленості та залученості здобувачів, візуальні інтерфейси, можливість виконання реальних завдань, створення власних проєктів сприяють активному навчанню.

Навчальні додатки стають частиною як формального освітнього процесу (уроки інформатики, заняття в університетах), так і неформального (онлайн-курси, самостійне навчання). Додатки відкривають нові горизонти для адаптивного навчання, інтеграції STEM-освіти, та підготовки до професійної діяльності у сфері ІТ. Узагальнюючи викладене, варто наголосити, що впровадження цифрових технологій у підготовку здобувачів з інформаційних технологій спричиняє трансформацію усталених методик навчання програмування та вимагає переходу до більш гнучких і технологічно орієнтованих освітніх моделей.

Використання спеціалізованих навчальних застосунків забезпечує індивідуалізацію навчальної траєкторії, підтримує можливість опанування матеріалу на різних типах пристроїв і розширює доступ до інтерактивного навчального контенту.

Гейміфікація, візуалізація складних понять та інтерактивні формати взаємодії підвищують рівень залученості й сприяють формуванню стійкої внутрішньої мотивації. Така модель навчання стимулює розвиток практичних умінь, критичного мислення й здатності до самостійної діяльності, що є важливими характеристиками сучасного фахівця у галузі програмування.

Доцільно підкреслити, що активне впровадження цифрових рішень супроводжується низкою об’єктивних викликів. Серед основних чинників, що потребують уваги, виокремлюються потреба у створенні високоякісних навчальних матеріалів, належний рівень цифрової компетентності педагогів та доступність відповідної технічної інфраструктури.

Попри наявні обмеження, сучасні тенденції розвитку цифрової освіти та зростання сегмента EdTech дають підстави прогнозувати подальше посилення впливу мобільних і веборієнтованих навчальних застосунків у системі підготовки фахівців, зокрема у сфері програмування.

Крім того, важливим є питання доступності, адже не всі навчальні додатки мають повну функціональність у безкоштовних версіях, що може обмежити можливості деяких користувачів. Також слід враховувати ризик

втрати мотивації при відсутності персонального підходу або підтримки з боку викладача. Надмірна гейміфікація без глибокого опрацювання матеріалу може створити хибне уявлення про рівень засвоєння знань.

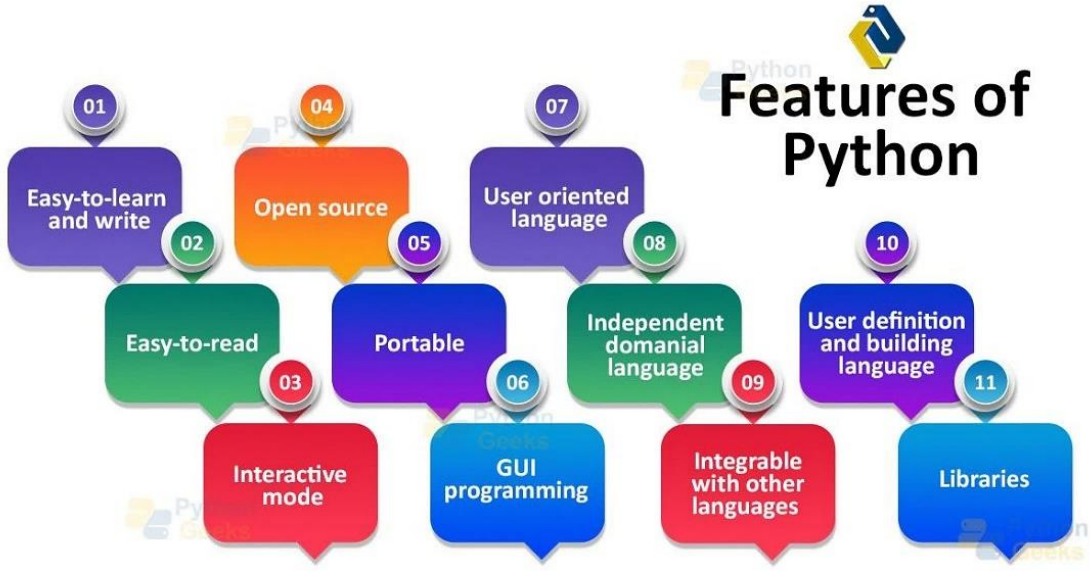

Рис. 1.7 Майбутнє з Python

У перспективі можна очікувати ширшого впровадження штучного інтелекту в навчальні додатки, що дозволить реалізовувати індивідуалізоване навчання з адаптацією під потреби конкретного учня. Також прогнозується розвиток інтегрованих освітніх платформ, які поєднуватимуть вивчення Python із практичними проектами, онлайн-змаганнями, автоматичним оцінюванням прогресу та спільнотами для підтримки користувачів. Розширення доступу до якісних цифрових ресурсів стане важливою складовою цифрової трансформації освіти.

Загалом, Python є не лише зручною мовою для старту, але й потужним інструментом для професійного розвитку, що робить її престижною та стратегічно важливою у сучасній IT-освіті.

Вивчення Python із використанням додатків є сучасним та ефективним способом здобуття програмістських навичок. Поєднання доступності, інтерактивності та практичного спрямування робить цей підхід надзвичайно привабливим для широкого кола користувачів. Надалі можна очікувати інтеграції ще більш інноваційних технологій у навчальний процес.

На основі проведеного дослідження можна зробити висновок, що вивчення Python за допомогою додатків є ефективним і актуальним підходом у підготовці фахівців різного профілю. Простота мови, потужна функціональність та активна спільнота розробників сприяють її широкому поширенню в освіті. Використання навчальних додатків дозволяє персоналізувати процес навчання, підвищити мотиваційну складову та створити сприятливе середовище для практичного засвоєння знань.

Застосування додатків у процесі вивчення Python відкриває можливості не лише для ефективного засвоєння програмування, а й для розвитку креативного мислення, самостійності та цифрової грамотності здобувачів. Особливої актуальності така методика набуває в умовах дистанційного та змішаного навчання, де важливо зберігати високий рівень залученості та взаємодії з навчальним матеріалом.

У перспективі інтеграція вивчення Python у орієнтованих додатках може стати одним з ключових напрямів цифрової трансформації. Важливо, щоб педагогічна спільнота продовжувала досліджувати та впроваджувати новітні інструменти, що відповідають вимогам часу та сприяють підвищенню якості освітніх послуг. Таким чином, мова Python виступає не лише засобом професійного розвитку, а й платформою для інноваційного навчання.

1.3 Godot як основний інструмент для розробки кросплатформенного додатку

Стрімкий розвиток цифрових технологій зумовлює підвищений інтерес до створення кросплатформених програмних рішень. Зростання кількості пристроїв і операційних систем формує потребу у програмному забезпеченні, яке здатне функціонувати на різних платформах без суттєвих змін програмного коду. Така тенденція орієнтує розробників на застосування технологій та інструментів, що спрощують процес розроблення, оптимізують продуктивність і забезпечують належну якість програмного продукту.

Серед таких інструментів особливе місце займає Godot Engine – безкоштовний, відкритий рушій для створення 2D та 3D додатків. З моменту свого першого випуску в 2014 році, Godot здобув популярність серед незалежних розробників та малих студій завдяки своїй гнучкості, потужності та активній спільноті користувачів. Можливості рушія дозволяють створювати як прості, так і складні проекти з використанням сучасних технологій та підходів.

«Godot – це безкоштовний ігровий механізм із відкритим вихідним кодом, незалежний від платформи, випущений за ліцензією MIT. Спочатку вона була розроблена лише для кількох латиноамериканських компаній, але після випуску стала модною серед користувачів відеоігор у всьому світі» [27].

Модульна архітектура двигуна та підтримка різних мов програмування, таких як GDScript, C# та C++, дозволяють гнучко адаптуватися до потреб проекту. Godot також пропонує вбудовані інструменти для роботи з графікою, фізикою, анімацією та мережею, що спрощує процес розробки та зменшує залежність від сторонніх бібліотек.

Завдяки активній спільноті та постійному оновленню документації, розробники можуть швидко знаходити відповіді на свої запитання та обмінюватися досвідом. Крім того, Godot підтримує експорт проектів на різні платформи з мінімальними змінами в коді, що робить рушій ідеальним вибором для кросплатформенної розробки. Перевагами рушія є повністю безкоштовна ліцензія MIT, що робить його привабливим для інді-розробників та освітніх установ. Godot має інтуїтивно зрозумілий інтерфейс та використовує GDScript – мову, схожу на Python, що спрощує навчання для новачків.

Unity пропонує розширений набір інструментів для оптимізації продуктивності та велику спільноту з багатою бібліотекою ресурсів, забезпечує передові графічні можливості та підходить для створення AAA-проєктів.

Godot Engine є потужним інструментом для розробки кросплатформних додатків, що підтримує експорт на різні платформи, включаючи настільні системи, мобільні пристрої, веб-платформи, а також платформи віртуальної та доповненої реальності, що дозволяє розробникам створювати додатки з єдиною кодовою базою, що значно спрощує процес розробки та обслуговування.

Godot надає користувачам повністю інтегроване середовище розробки ігор, що дозволяє розробникам створювати гру з нуля, використовуючи лише інструменти для створення вмісту, наприклад музику, зображення чи текстури. Ігрові компоненти – від кодів до графічних ресурсів - зберігаються у файловій системі комп'ютера (замість бази даних). Режим зберігання призначений для полегшення спільної роботи команд розробників ігор над вихідним кодом за допомогою відстеження версій [27].

«У сфері розробки відеоігор методологія створення ігор з часом значно змінилася. Ще у 1980-х роках кілька компаній почали використовувати власні двигуни. Наприклад, у випадку з Super Mario Bros (1985) Nintendo використовувала ігровий движок із швидким прокручуванням, який спочатку був розроблений для Excitebike (1984)» [28]. «Концепція ліцензування ігрових движків іншим компаніям як самостійного продукту існує з середини 90-х років. У той час такі ігри, як Doom (1993), розроблялися з використанням ігрового движка id Tech» [28, с. 181].

Однією з ключових особливостей Godot є використання власної мови програмування GDScript, яка синтаксично схожа на Python, що забезпечує швидке прототипування та легкість у навчанні для новачків . Крім того, підтримка C# та візуального скриптингу через VisualScript розширює можливості для розробників з різним рівнем підготовки та уподобаннями .

Архітектура Godot побудована на концепції сцени та вузлів, що дозволяє створювати модульні та повторно використовувані компоненти. Кожен вузол може мати свої власні властивості та поведінку, що сприяє організації коду та спрощує процес розробки складних додатків .

Вбудовані засоби для роботи з 2D та 3D графікою, включаючи фізичні рушії, систему анімації та підтримку шейдерів, дозволяють створювати візуально привабливі та функціональні додатки. Зокрема, наявність вбудованого фізичного рушія спрощує реалізацію реалістичної фізики в додатках [27]. «Godot Engine – це все-в одному кросплатформенний ігровий движок, яким можна користуватися безкоштовно» [27]. Наразі механізм Godot підтримує наступні платформи: вікна, macOS, Linux, Android, iOS, BlackBerry 10, FreeBSD, OpenBSD / DragonFly BSD, HTML5, Windows Runtime (WinRT), Універсальна платформа Windows (UWP).

Щодо розробки мобільних додатків для iOS та Android, Godot надає функціональні інструменти для кросплатформенного експорту. Рушій підтримує експорт на обидві платформи, хоча для iOS потрібна компіляція на macOS. Godot дозволяє створювати легкі та ефективні мобільні додатки, особливо у 2D-сегменті. Однак, у порівнянні з Unity, який має більш зрілу екосистему та розширені можливості для мобільної оптимізації, Godot може вимагати більше ручної роботи для досягнення високої продуктивності на різних пристроях. Тим не менш, для невеликих та середніх проектів Godot є конкурентоспроможним вибором.

Godot Engine, попри свою відкриту ліцензію та орієнтацію на інді-розробників, став основою для низки комерційно успішних ігор, які продемонстрували потенціал у створенні якісних продуктів. Одним із найяскравіших прикладів є Buckshot Roulette – інноваційна гра, що поєднує елементи хорору та карткової стратегії. Розроблена М. Клубнікою, вона вийшла на платформі Steam у квітні 2024 року та до кінця року продала понад 4 мільйони копій, отримавши схвальні відгуки від критиків за оригінальний геймплей та атмосферу.

Іншим прикладом є Cruelty Squad – сюрреалістичний шутер від першої особи з унікальним візуальним стилем, який здобув популярність серед геймерів за свій нестандартний підхід до дизайну та геймплею . Також варто згадати Cassette Beasts – покемоноподібну RPG з відкритим світом, яка

отримала позитивні відгуки за інноваційні механіки та ретро-естетику. Проєкти демонструють, що Godot здатен підтримувати розробку не лише інді-ігор, але й комерційно успішних продуктів з широким охопленням аудиторії.

У підсумку, Godot Engine є гнучким інструментом для розробки кросплатформних додатків, що поєднує в собі широкий спектр можливостей та простоту використання. Відкритий код та активна спільнота сприяють постійному розвитку та вдосконаленню рушія, забезпечуючи актуальність та відповідність сучасним вимогам індустрії. Можливість експорту на різні платформи з єдиної кодової бази значно спрощує процес розробки та обслуговування додатків, що є важливим фактором у світі інформаційних технологій.

Таким чином, використання Godot Engine як основного інструменту для розробки кросплатформних додатків є обґрунтованим вибором, що поєднує в собі ефективність та економічну доцільність. Рушій дозволяє створювати якісні та функціональні додатки, що відповідають сучасним стандартам та вимогам користувачів.

Висновок до першого розділу

Проведений аналіз засвідчує, що кросплатформенність стала ключовим фактором у сучасній цифровій екосистемі, особливо у сфері розробки програмного забезпечення для освіти. Вона забезпечує універсальний доступ до програмних рішень, дозволяючи охопити широку аудиторію користувачів незалежно від операційної системи або пристрою. Такий підхід сприяє зниженню витрат, спрощенню обслуговування додатків, уніфікації досвіду користувача та підвищенню ефективності команд розробників. Поява потужних фреймворків - таких як Flutter, React Native, Xamarin, Electron, Qt – відкриває нові горизонти для швидкої та ефективної реалізації кросплатформенних рішень.

Особливу роль у цьому процесі відіграє Python – мова, яка поєднує простоту, універсальність і потужний набір інструментів. Її відкритий синтаксис, гнучка логіка та широка підтримка з боку спільноти роблять Python ідеальним вибором для початкового вивчення програмування. Водночас вона зберігає свою актуальність і в професійному середовищі, активно використовуючись у сферах штучного інтелекту, аналізу даних, розробки веб- та мобільних додатків. Python не лише знижує поріг входження у світ програмування, а й дозволяє будувати реальні, практично значущі проєкти – що особливо важливо у контексті сучасної STEM-освіти.

Інтеграція Python з навчальними додатками створює новий освітній простір, де навчання стає гнучким, адаптивним, гейміфікованим і глибоко інтерактивним. Різноманітні освітні платформи (як-от Replit, SoloLearn, Codecademy, Coursera, CodeCombat) дозволяють індивідуалізувати навчання, забезпечити миттєвий зворотний зв’язок, розвивати критичне мислення та мотивувати до самостійного вивчення. Такі інструменти відіграють важливу роль у трансформації освітніх підходів, орієнтуючи здобувачів не лише на теоретичне засвоєння знань, а й на практичну діяльність.

Водночас особливу увагу заслуговує Godot Engine – сучасний, відкритий інструмент для кросплатформенної розробки ігор та додатків. Підтримка мов програмування (особливо GDScript, що нагадує Python), модульна структура та можливість експорту на різні платформи роблять рушій ідеальним рішенням для навчальних проєктів, інді-розробників та малих студій. Приклади успішних ігор, розроблених у Godot (Cruelty Squad, Buckshot Roulette, Cassette Beasts), демонструють, що рушій здатен забезпечити високий рівень реалізації навіть для комерційних продуктів.

Таким чином, поєднання Python, кросплатформенних технологій і таких інструментів, як Godot, створює потужне підґрунтя для ефективної підготовки фахівців у сфері інформаційних технологій. Це дозволяє забезпечити доступну, гнучку та актуальну IT-освіту, що відповідає викликам цифрової епохи, сприяє формуванню компетентностей XXI

століття та відкриває шлях до професійного успіху в умовах глобальної технологічної трансформації.

РОЗДІЛ 2
СТРУКТУРА РОЗРОБКИ КРОСПЛАТФОРМЕННОГО ДОДАТКУ ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ PYTHON

2.1 Розробка адаптивного дизайну під різні додатки

Розробка кросплатформених додатків для навчання програмуванню на мові Python передбачає комплексне поєднання технічних, методичних та педагогічних аспектів. Такий підхід дозволяє створювати програмні продукти, здатні функціонувати на різних операційних системах та пристроях, забезпечуючи доступність і гнучкість освітнього процесу.

Структура розробки включає визначення цільової аудиторії та навчальних завдань, проєктування інтерфейсу та архітектури додатку, вибір відповідних технологій і бібліотек, а також інтеграцію інтерактивних та практико-орієнтованих елементів навчання. Така системність забезпечує ефективне поєднання навчальної цінності з технологічною функціональністю, що сприяє формуванню практичних навичок програмування та розвитку цифрової компетентності здобувачів освіти.

Тому, першим етапом практичної розробки адаптивного дизайну стало створення макетів інтерфейсу для різних платформ. На цьому етапі були визначені ключові екрани додатка – головне меню, навчальні модулі та екран виконання коду – та опрацьовані їхні взаємодії для забезпечення логічної та інтуїтивно зрозумілої навігації.

Особлива увага приділялася врахуванню різних розширень дисплеїв мобільних пристроїв та десктопів, щоб елементи управління залишалися доступними, зручними та візуально пропорційними. Завдяки такому підходу забезпечується узгодженість користувацького досвіду на різних платформах, збереження функціональності всіх компонентів та підтримка високого рівня

користувацької взаємодії, що є ключовим фактором ефективності навчального додатку.

Для оптимізації макетів використовувалися різні принципи розташування елементів: сіткові системи, відступи та масштабування шрифтів, що дозволяє забезпечити однаковий користувацький досвід незалежно від розмірів екрану. Особлива увага приділялася розташуванню кнопок для запуску коду та навігаційних елементів, щоб вони були легкодоступними навіть на невеликих екранах смартфонів.

Наступним кроком стало тестування попередніх макетів із симуляцією різних розмірів дисплеїв у Godot, що дало можливість оцінити масштабування тексту, кнопок і панелей, а також перевірити зручність інтерфейсу при зміні орієнтації екрану. Результати цього тестування стали основою для наступного підпункту – реалізації масштабування та адаптивних компонентів інтерфейсу.

Після створення макетів наступним етапом було налаштування системи масштабування інтерфейсу у Godot, для чого використовувалися функції Control nodes та Container nodes, які дозволяють автоматично підлаштовувати елементи під розміри вікна або екрану. Головним завданням цього етапу є забезпечення коректного відображення елементів інтерфейсу – кнопок, панелей та текстових полів – на різних типах пристроїв із запобіганням перекриттю елементів або появі непотрібних проміжків.

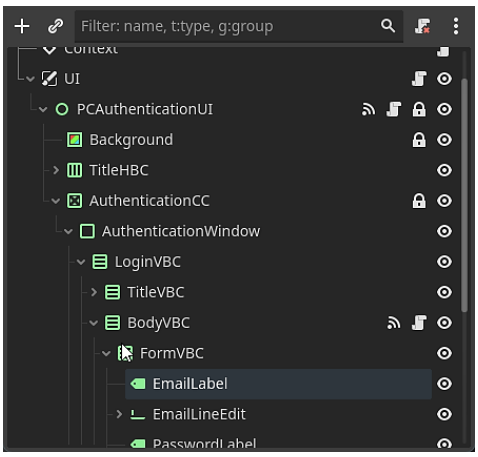

Рис. 2.1 Ієрархія сцени авторизації

Особлива увага приділялася пропорційному масштабуванню елементів інтерфейсу. Використовувалися анкорні точки та відсоткові відступи, щоб кнопки та панелі зберігали співвідношення розмірів на різних екранах. Подібні налаштування дозволяють уникнути ситуацій, коли елементи стають надто маленькими на великих екранах або навпаки – займають занадто багато місця на мобільних пристроях.

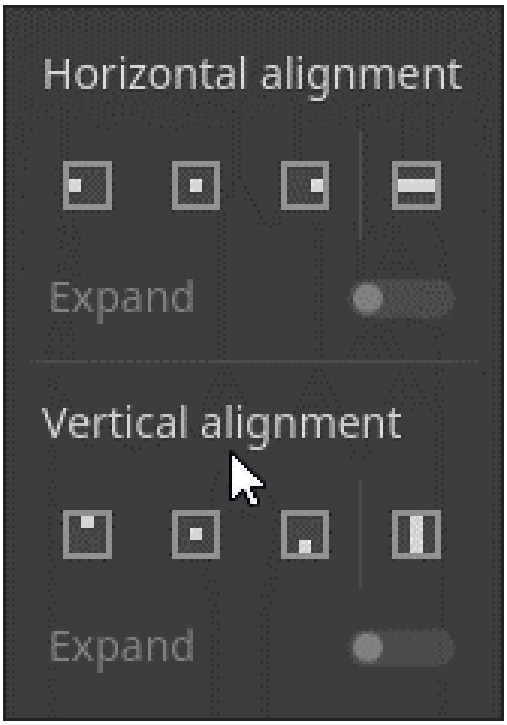

Рис. 2.2 Панель вирівнювання інтерфейсу

Для діагностики ефективності системи масштабування проводилося тестування на різних резолюціях та орієнтаціях екранів. Виявлені проблеми з масштабуванням були виправлені через додаткове налаштування Margins і Size Flags, що забезпечило більш плавну адаптацію інтерфейсу під різні платформи, що дозволило переходити до наступного етапу – реалізації адаптивних компонентів UI.

Після налаштування системи масштабування наступним кроком стала реалізація адаптивних компонентів інтерфейсу, а саме: кнопки, панелі навігації, поля введення та елементи відображення результатів виконання коду. Кожен елемент був налаштований так, щоб автоматично

підлаштовуватися під розмір екрану і змінювати свою форму або розташування у залежності від доступного простору.

Для забезпечення гнучкості інтерфейсу використовувалися контейнери типу VBoxContainer, HBoxContainer та GridContainer. Вони дозволяють компонувати елементи у вертикальні та горизонтальні блоки, зберігаючи пропорції та відступи при зміні розміру екрану. Такий підхід значно спрощує додавання нових елементів у майбутньому та забезпечує єдиний стиль інтерфейсу для всіх платформ.

На фінальному етапі проводилося тестування адаптивності компонентів на різних пристроях та симуляторах. Було перевірено, як змінюється розмір кнопок, панелей і текстових полів, а також чи не виникають конфлікти при взаємному перекритті елементів. Результати тестування дозволили скорегувати деякі параметри Size Flags і Margins, забезпечивши плавну та коректну адаптацію інтерфейсу під усі цільові платформи.

Після реалізації адаптивних компонентів наступним кроком стала оптимізація розташування елементів на різних типах пристроїв. Основна мета цього етапу – забезпечити зручність користування додатком на мобільних телефонах із невеликими екранами та на десктопах із великими дисплеями. Було проведено коригування позицій панелей навігації, кнопок запуску коду та полів введення для досягнення гармонійного та інтуїтивно зрозумілого інтерфейсу.

Для оптимізації використовувалися різні методи: змінні розміри елементів, динамічне відображення деяких панелей лише на великих екранах, а також групування схожих елементів у контейнерах, що дозволяє зменшити навантаження на екран мобільного пристрою і забезпечити логічну структуру інтерфейсу на десктопі, де є більше простору для додаткових елементів.

На завершальному етапі проводилося тестування оптимізованого розташування елементів на різних пристроях, що дозволило виявити невеликі перекриття або проблеми з відступами та скоригувати їх. В результаті

інтерфейс став більш зрозумілим і комфортним для користувача незалежно від типу пристрою, що підвищує загальну якість користувацького досвіду.

Після завершення налаштувань та оптимізації інтерфейсу наступним етапом стало тестування адаптивності на різних форма-факторах. Для цього додаток запускали на різних мобільних пристроях, планшетах і десктопних комп’ютерах, а також використовували емулятори різних розмірів екранів. Відповідно, основна мета тестування – перевірити правильність відображення всіх елементів UI та їхню функціональність при зміні роздільної здатності або орієнтації екрана.

Під час тестування особливу увагу приділяли ключовим компонентам – кнопкам запуску коду, полям для введення та панелям навігації. Перевірялися взаємодії між елементами, правильність масштабування та збереження пропорцій. У разі виявлення проблем відразу проводилися коригування анкорів, відступів та параметрів контейнерів для забезпечення стабільного та комфортного відображення інтерфейсу.

Фінальний етап тестування включав перевірку адаптивності при зміні орієнтації екрана та переході між різними розділами додатка. Результати показали, що елементи інтерфейсу залишаються зручними та функціональними на всіх платформах, а користувацький досвід не залежить від розміру пристрою, що дозволило завершити підрозділ розробки адаптивного дизайну та перейти до інтеграції додатка з серверним застосунком Firebase.

У підсумку проведені заходи з розробки адаптивного дизайну забезпечили формування інтерфейсу, здатного коректно функціонувати на широкому спектрі пристроїв із різними характеристиками. Впровадження системи масштабування, створення адаптивних компонентів та оптимізація їх просторової організації сприяли підвищенню ергономічності та доступності користувацького середовища. Отримані результати засвідчили ефективність обраних підходів та дозволили сформувати стійку основу для подальшого розвитку програмного рішення.

2.2 Інтеграція Godot з серверним застосунком Firebase

Інтеграція ігрового або навчального рушія Godot із серверним застосунком Firebase відкриває нові можливості для створення кросплатформених програмних продуктів із розширеною функціональністю. Поєднання Godot, який забезпечує гнучке та ефективне проєктування інтерактивного інтерфейсу та логіки додатків, з Firebase, що пропонує масштабовані сервіси для зберігання даних, автентифікації користувачів та реального часу, дозволяє створювати адаптивні системи з підтримкою хмарних технологій.

Такий підхід сприяє підвищенню інтерактивності, автоматизації обробки даних і спрощує управління користувацькими сценаріями, що особливо важливо для навчальних, ігрових та комерційних застосунків. Інтеграція Godot та Firebase забезпечує комплексне поєднання клієнтської та серверної частини, оптимізує процес розробки та підвищує ефективність взаємодії з користувачем у реальному часі.

Firebase – це платформа розробки від Google, яка надає набір хмарних сервісів для створення веб- та мобільних додатків. Основне призначення Firebase – спрощення процесу розробки, зберігання даних, керування користувачами та аналітики без необхідності самостійно розгортати серверну інфраструктуру. Отже, ключові компоненти Firebase:

- Analytics та Crashlytics – інструменти для моніторингу використання додатку та відстеження помилок;
- Authentication – система автентифікації користувачів із підтримкою різних методів (email, Google, Facebook, телефон тощо);
- Cloud Functions – серверні функції, що виконуються у хмарі у відповідь на події додатку або бази даних;
- Cloud Storage – хмарне сховище для зберігання файлів, зокрема зображень, відео та документів;

- Firebase Hosting – хостинг для веб-додатків із підтримкою HTTPS і глобальної доставки контенту;
- Realtime Database та Firestore – бази даних у хмарі, що дозволяють зберігати та синхронізувати дані в реальному часі між користувачами і пристроями.

Firebase популярний серед розробників завдяки простоті інтеграції, масштабованості та можливості швидко створювати кросплатформені рішення без складного налаштування серверів.

Тому, першим етапом інтеграції серверного застосунку Firebase стало підключення відповідного аддону до проєкту в Godot. Для цьогобуло завантажено та встановлено Godot Firebase Addon у робочий проєкт, після чого налаштувано основні параметри підключення, включно з API Key, Project ID та іншими ідентифікаторами Firebase, що дозволяє забезпечити подальший обмін даними між клієнтським додатком та хмарним сервером.

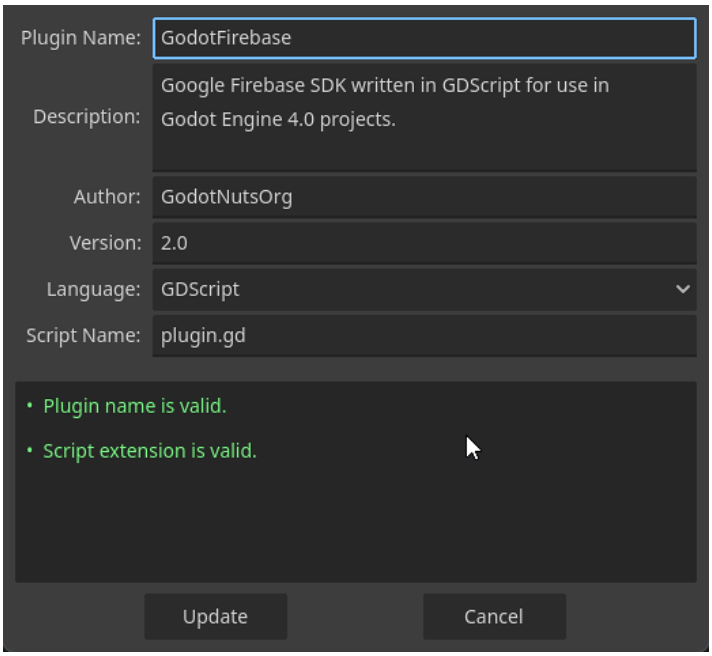

Рис. 2.3 Імпорт розширення Python

Наступним етапом стало підключення до різних сервісів Firebase, таких як Authentication, Realtime Database та Firestore. Було створено конфігураційний файл для збереження параметрів підключення та перевірено зв’язок між додатком та сервером через прості запити на читання та запис

даних, що дозволяє переконатися, що клієнтський додаток коректно підключається до хмарного середовища і готовий до подальшої інтеграції.

На завершальному етапі проводилося тестування підключення на різних платформах, включно з мобільними та десктопними пристроями. Було перевірено стабільність з'єднання та швидкість обміну даними, що дозволило виявити та усунути початкові проблеми з конфігурацією. Після цього додаток був готовий до реалізації функцій авторизації та зберігання даних користувача.

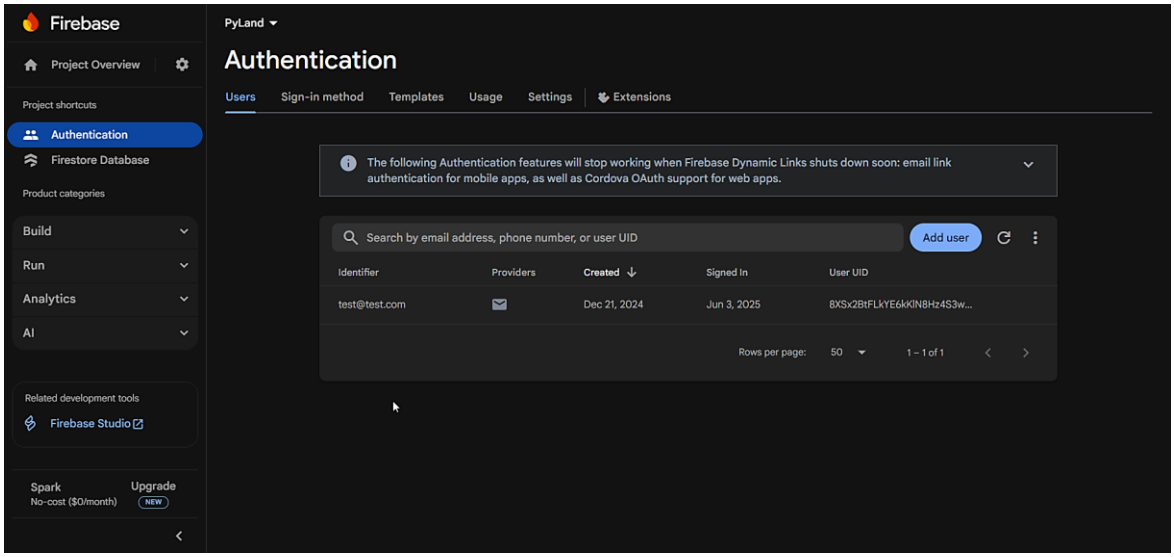

Рис. 2.4 Панель керування Firebase

Після підключення Firebase до проєкту наступним етапом стало налаштування авторизації користувачів через Firebase Authentication. Для цього були реалізовані базові форми входу та реєстрації з використанням електронної пошти та пароля. Кожна форма підключалася до відповідних методів Firebase, що дозволяє створювати нові акаунти та перевіряти існуючі при вході.

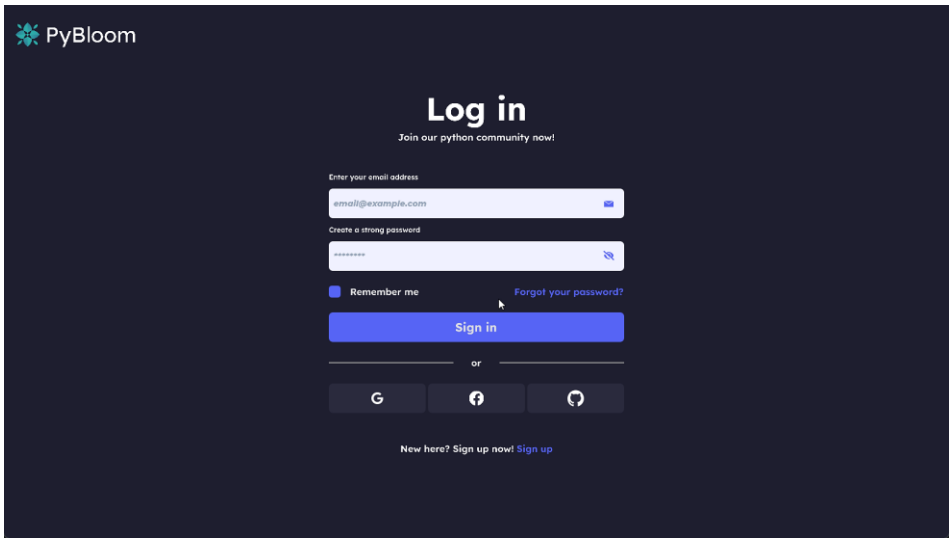

Рис. 2.5 Меню авторизації

Для підвищення зручності користувачів також була реалізована можливість відновлення пароля та перевірки автентичності через електронну пошту. При авторизації додаток перевіряє правильність введених даних та повертає повідомлення про помилки, що забезпечує коректну роботу системи безпеки. Усі дії обробляються через Godot Firebase Addon, що забезпечує швидкий та стабільний обмін інформацією між клієнтом і сервером.

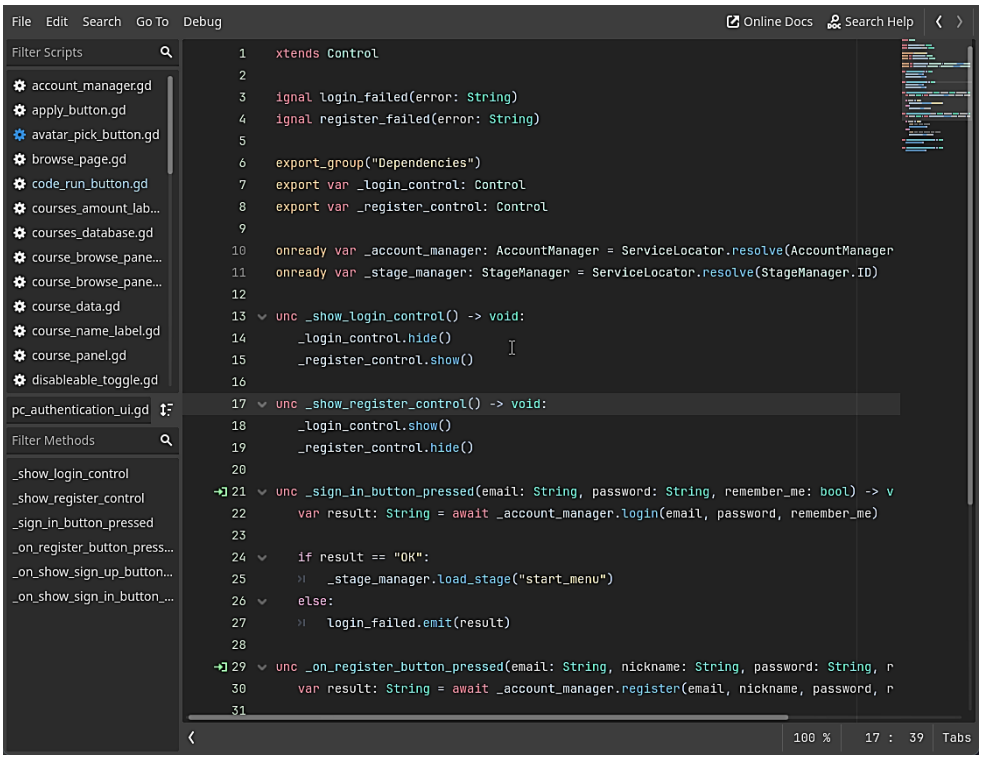

Рис. 2.6 Скрипт авторизації

На наступному етапі проводилося тестування авторизації на різних пристроях. Було перевірено реєстрацію, вхід, відновлення пароля та вихід з акаунта, що дозволило виявити та усунути проблеми з синхронізацією даних і забезпечити стабільну роботу користувацької системи додатка. Після успішного налаштування авторизації можна переходити до реалізації функцій читання та запису даних користувача у базі Firebase.

Після реалізації авторизації наступним етапом стало налаштування механізмів читання та запису даних користувача у Firebase. Для цього у Godot використовувалися методи Godot Firebase Addon, що дозволяють взаємодіяти як з Realtime Database, так і з Firestore. Було створено базові функції для запису прогресу користувача, збереження налаштувань додатка та результатів виконання навчальних завдань.

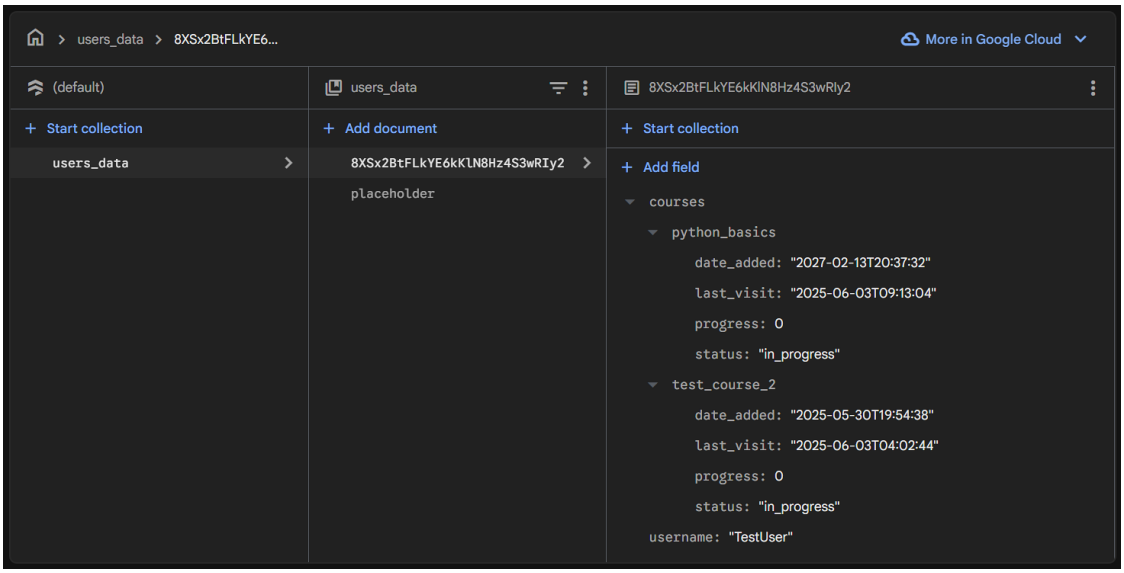

Рис. 2.7 Структура даних користувача

Для читання даних реалізовані функції запиту інформації про користувача під час входу в додаток та під час переходу між навчальними модулями. Дані з Firebase автоматично підвантажуються в інтерфейс додатка, що дозволяє відображати актуальний прогрес та результати виконаних завдань. Також була налаштована обробка помилок при втраті зв’язку або некоректних запитах.

```gdscript
func _load_courses() -> bool:
    var user_data: Dictionary = await _account_manager.get_data()
    if user_data == {}: return false

    var _courses: Dictionary[String, CourseData] = _courses_database.get_courses()
    for course_id: String in _courses:
        var course_panel: CourseBrowsePanel = _course_browse_panel_prefab.instantia
        _course_panels_parent.add_child(course_panel)

        var status: String
        if user_data.courses.has(course_id):
            status = user_data.courses[course_id].status
        else:
            status = "available"

        var course_data: CourseData = _courses[course_id]
        course_panel.set_data(
            course_id,
            course_data.get_banner(),
            course_data.get_display_name(),
            course_data.get_lessons_amount(),
            status
        )
```

Рис. 2.8 Скрипт завантаження курсів

На фінальному етапі проводилося тестування механізмів запису та читання даних на різних платформах. Перевірялися швидкість синхронізації, коректність збереження прогресу та відновлення даних при повторному вході користувача, що дозволило переконатися в стабільній роботі клієнт-серверної взаємодії та готовності додатку до наступного етапу – синхронізації локальних даних із хмарним сховищем Firebase.

Наступним етапом після налаштування читання та запису даних стало забезпечення синхронізації локальних даних додатка з Firebase. Для цього була реалізована логіка автоматичного оновлення даних при зміні прогресу користувача або налаштувань додатка. Зміни в локальних даних відразу передавалися на сервер, а оновлені дані з Firebase підвантажувалися в додаток для відображення актуальної інформації.

Особливу увагу приділено вирішенню конфліктів при одночасному редагуванні даних на різних пристроях. Було реалізовано механізм перевірки часових міток останніх змін, що дозволяє визначати актуальні дані та уникати їх перезапису. Такий підхід забезпечує цілісність інформації та стабільну роботу додатку незалежно від кількості пристроїв, на яких користувач заходить у свій акаунт.

На фінальному етапі проводилося тестування синхронізації на різних платформах та мережевих умовах. Перевірялися коректність оновлення даних, відновлення прогресу після перезапуску додатка та робота в умовах нестабільного інтернет-з’єднання. Результати показали, що дані користувача зберігаються та синхронізуються коректно, що забезпечує безперебійну взаємодію між клієнтським додатком і хмарним сховищем Firebase.

Після реалізації синхронізації локальних та хмарних даних наступним етапом стало налагодження та тестування обміну інформацією між клієнтським додатком і Firebase. Для цього було проведено серію тестів на різних платформах, включно з мобільними пристроями та десктопами, з метою перевірки стабільності з’єднання та коректності обробки запитів.

Особлива увага приділялася відстеженню помилок при записі та читанні даних, перевірці обробки нестабільного або відсутнього інтернет-з’єднання, а також контролю черг запитів для уникнення конфліктів. Було реалізовано логування основних подій обміну даними, що дозволяє швидко виявляти та усувати проблеми у взаємодії між додатком та сервером.

Наступним тестування перевірялися усі сценарії користувацької взаємодії з даними: реєстрація, авторизація, запис та читання прогресу, відновлення даних після повторного входу. Результати показали стабільну роботу клієнт-серверної взаємодії на всіх платформах, що дозволило підтвердити готовність додатку до реалізації інтерактивного Python-середовища.

Інтеграція Godot із серверним застосунком Firebase забезпечила стабільну інфраструктуру для обміну, зберігання та синхронізації даних користувачів у реальному часі. Реалізовані механізми автентифікації, доступу до баз даних та обробки запитів створюють надійний зв’язок між клієнтським додатком і хмарним сервером. Використані підходи дозволяють підтримувати функціональність додатку на різних платформах і забезпечують високий рівень стабільності та ефективності роботи програмного середовища.

Узагальнюючи проведену роботу, інтеграція рушія Godot із серверним застосунком Firebase продемонструвала ефективний підхід до створення кросплатформених додатків із розширеною функціональністю. Поєднання Godot та Firebase дозволило забезпечити комплексне управління клієнтською та серверною частинами, що включає автентифікацію користувачів, зберігання та синхронізацію даних у реальному часі, а також обробку запитів із високою стабільністю та швидкістю.

Реалізація базових механізмів авторизації, читання та запису даних у Realtime Database та Firestore забезпечила надійну взаємодію користувача з додатком, а налаштування синхронізації локальних і хмарних даних дозволило підтримувати цілісність інформації та актуальний стан прогресу на різних платформах. Проведене тестування на мобільних і десктопних пристроях підтвердило стабільність підключення, коректність обробки даних та готовність додатку до використання інтерактивного середовища Python.

Отримані результати свідчать, що інтеграція Godot із Firebase забезпечує надійну інфраструктуру для кросплатформених освітніх і навчальних додатків, підвищує ефективність взаємодії користувача з програмним продуктом і створює передумови для подальшого розширення функціоналу, включно з реалізацією складних навчальних сценаріїв, аналізом результатів та інтеграцією додаткових сервісів.

2.3 Вбудування інтерпретатора Python для створення начального інтерактивного середовища

Вбудування інтерпретатора Python у програмні додатки відкриває нові можливості для створення інтерактивного навчального середовища, орієнтованого на початкове освоєння мови програмування. Такий підхід дозволяє користувачам безпосередньо виконувати код, експериментувати з структурами та отримувати миттєвий зворотний зв’язок, що значно підвищує ефективність навчання та формує практичні навички програмування.

Інтеграція інтерпретатора в середовище додатку забезпечує поєднання навчальної цінності з функціональністю програмного продукту, дозволяє реалізувати адаптивні сценарії виконання завдань та створює умови для поступового розвитку алгоритмічного мислення здобувачів освіти.

Для створення інтерактивного середовища виконання Python-коду всередині додатку було застосовано механізм GDExtension, який забезпечує можливість розширення функціональності Godot за рахунок підключення зовнішніх модулів. Такий підхід дозволяє інтегрувати сторонні інтерпретатори та організувати їхню взаємодію з ігровим рушієм без внесення змін у ядро, що підвищує гнучкість і безпеку розробки.

Основним завданням цього етапу стало розроблення розширення, здатного виконувати Python-скрипти, обробляти їхній результат та передавати назад до клієнтської частини додатку. При цьому були передбачені механізми контролю виконання, обробки помилок і ізоляції середовища, що забезпечує стабільність роботи рушія та безпечну взаємодію користувацького коду з системними компонентами програми.

У ході інтеграції було реалізовано структуру GDExtension-модуля, що забезпечує прийом викликів із Godot та їх передачу в вбудований Python-інтерпретатор. Процес включав визначення інтерфейсів, реєстрацію класів і розробку методів для виконання коду та обробки потенційних помилок. Особлива увага приділялася ізоляції середовища виконання, що гарантує безпечну роботу користувацького коду без впливу на внутрішні механізми додатку

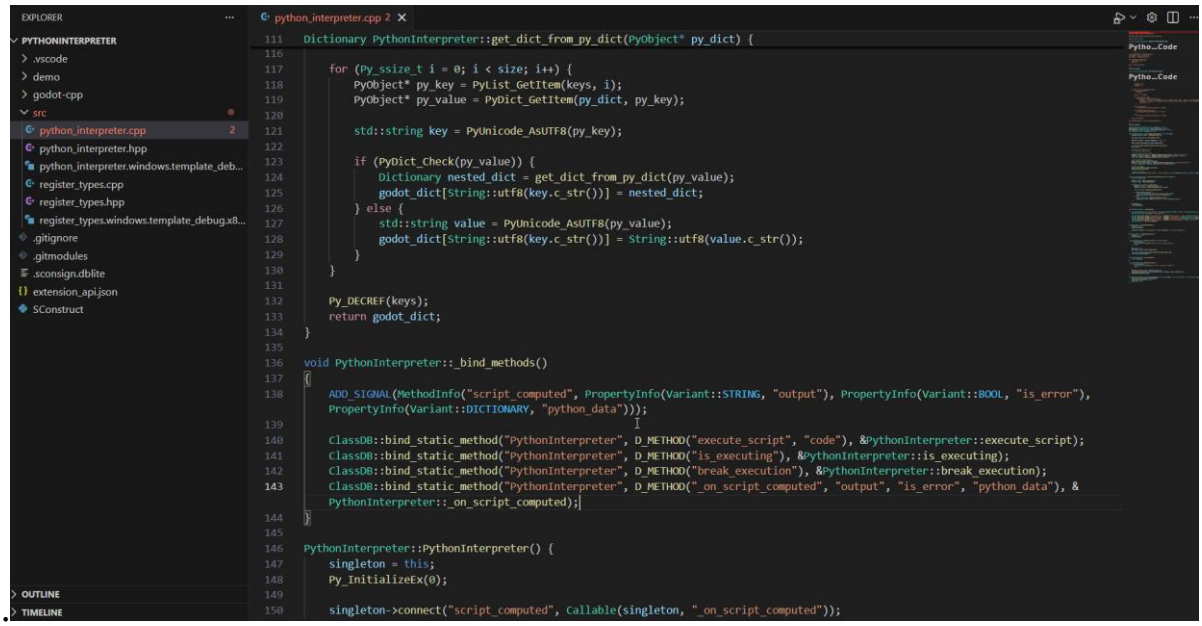

Рис. 2.9 Інтеграція GDExtension для Python

Після первинного налаштування інтеграції здійснили базове тестування, яке охоплювало виконання простих команд, обробку синтаксичних помилок та відображення результатів у вигляді текстових повідомлень, що підтвердило правильну роботу інтерпретатора та здатність обробляти виклики та забезпечило надійну основу для подальшої розробки повноцінного інтерактивного навчального середовища.

Інтегрувавши Python-інтерпретатор за допомогою GDExtension, наступним етапом стало впровадження функціоналу, що дозволяє додатку виконувати Python-код, введений користувачем. У рамках GDExtension-модуля були розроблені методи, які приймають текстові рядки, передають їх у вбудований інтерпретатор та повертають результати виконання у форматі, сумісному з Godot. Запроваджений механізм став базою для інтерактивних вправ, що дають змогу користувачеві одразу отримувати зворотний зв’язок на введений код.

Розробка функцій включала обробку стандартного потоку виведення Python, ізоляцію помилок та забезпечення коректного парсингу результатів. Особливе значення мала можливість перехоплювати як успішний вивід, так і виняткові ситуації, щоб інтерфейс міг відобразити їх користувачеві у

зручному форматі, що дозволило наблизити роботу інтерпретатора до поведінки реального середовища програмування.

```
R"(

import sys
class CustomStdOutErr:
    def __init__(self):
        self.buffer = ''
    def write(self, txt):
        self.buffer += txt
custom_std_out_err = CustomStdOutErr()
sys.stdout = custom_std_out_err
sys.stderr = custom_std_out_err

def redirected_input(promt):
    print(promt, end = '')
    return "0"

input = redirected_input

)";
```

Рис. 2.10 Налаштування буферів входу/виходу для Python

Додатково було реалізовано базовий шар обгортки у вигляді Godot-скриптів, які викликають функції GDExtension і передають результати елементам інтерфейсу. Саме на цьому етапі визначалися механізми синхронного та асинхронного виконання, що необхідно для забезпечення плавної роботи додатка та уникнення блокування головного потоку.

Для забезпечення інтерактивного навчального процесу було створено інтерфейс, що дозволяє користувачеві вводити та виконувати власний Python-код безпосередньо всередині додатку. Ядром цього інтерфейсного модуля став спеціалізований текстовий редактор, побудований на базі стандартного елемента Godot TextEdit, який було адаптовано для зручного та ефективного введення програмного коду.

Для підвищення зручності та наближення до функціональності мінімальної IDE були реалізовані додаткові можливості: автоматичний перенос рядків, підсвічування активного рядка, підтримка фіксованих відступів і базова обробка синтаксису Python. Крім того, передбачено

інтегроване відображення результатів виконання коду у спеціальній області виводу, що забезпечує миттєвий зворотний зв’язок користувачу. Така організація інтерфейсу дозволяє поєднати навчальні цілі з технічною зручністю, сприяє формуванню практичних навичок програмування та підвищує мотивацію здобувачів через інтерактивну взаємодію з додатком.

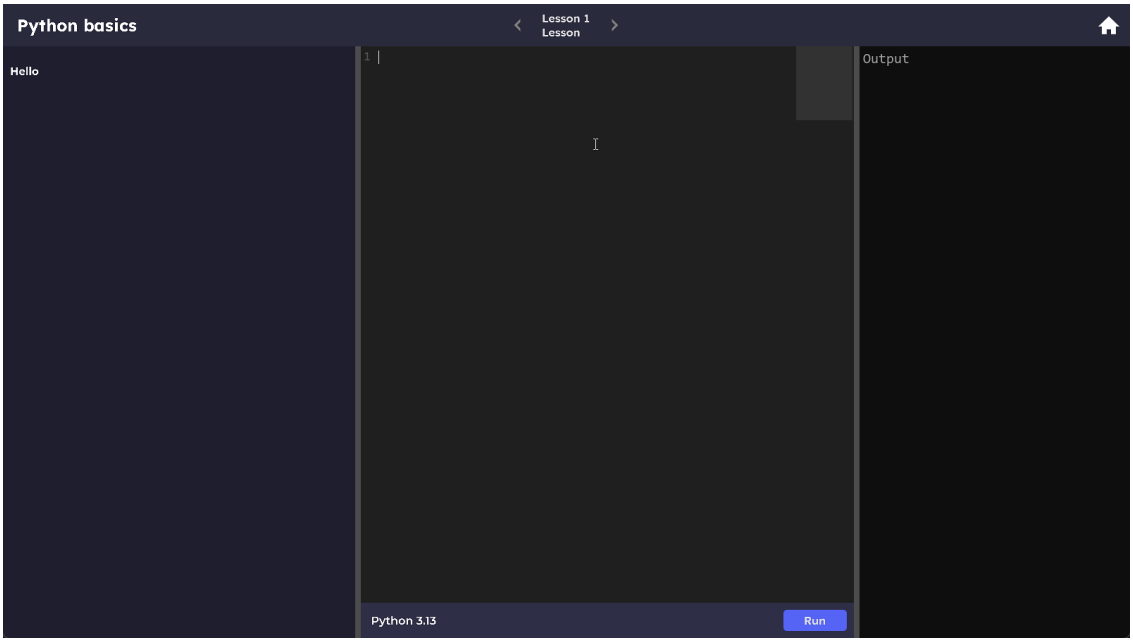

Рис. 2.11 Меню проходження курсу

Другим елементом інтерфейсу стала панель управління запуском коду. Вона містила кнопку виконання, а також реагувала на комбінацію клавіш, що пришвидшує взаємодію. Натискання на кнопку ініціює передачу введеного фрагмента коду до GDExtension, де відбувається обробка інтерпретатором Python. Важливо було забезпечити однозначність зв’язку між UI та внутрішнім модулем виконання, тому виклики було організовано через окремий контролерний клас.

Окрему увагу приділено структурі відображення помилок користувацького вводу. Під редактором було створено інформаційну область, де з’являються результати виконання або повідомлення про помилки, що дозволяє користувачеві одразу бачити, як працює код, та швидко вносити виправлення. Поділ інтерфейсу на окремі модулі – редактор,

панель взаємодії та зона результатів – створює логічно завершене та зрозуміле навчальне робоче середовище.

Після отримання результатів від інтерпретатора Python наступним кроком є їх коректна обробка та відображення у користувацькому інтерфейсі. Результати повертаються із GDExtension у вигляді текстових повідомлень, які можуть містити як стандартний вивід (stdout), так і інформацію про помилки (stderr). На виділеному етапі важливо було створити механізм, який здатний відокремити ці два типи повідомлень, щоб користувач отримував чіткий та структурований зворотний зв’язок.

Для відображення результатів була створена спеціальна панель виводу, яка автоматично оновлюється після завершення виконання коду. Якщо інтерпретатор повернув коректний результат, він виводиться у форматі звичайного тексту, а якщо були помилки – текст підсвічується іншим стилем або кольором. Такий підхід забезпечує швидке розуміння користувачем стану виконання програми. Також було реалізовано очищення попереднього виводу при кожному новому запуску, щоб уникати накопичення неактуальної інформації.

Під час обробки результатів окрему увагу приділено форматуванню виводу. Деякі Python-команди генерують багаторядкові або структуровані результати, тому текстова панель повинна коректно відображати їх без спотворення структури. Для цього були налаштовані параметри переносу рядків, вертикального скролінгу та автоматичної адаптації висоти блоку виводу. Завдяки цьому користувацький інтерфейс залишається зручним незалежно від складності результатів, які генерує Python-код.

Після реалізації механізмів виконання коду та інтеграції з інтерфейсом постала необхідність перевірити стабільність роботи інтерактивного Python-середовища в реальних умовах. Тестування включало запуск різних типів фрагментів коду: простих арифметичних виразів, циклів, рекурсивних функцій і невеликих скриптів, які могли тривати кілька секунд. Такий підхід

дозволив оцінити, як система обробляє як легкі, так і більш навантажувальні сценарії.

Другим напрямком тестування стала перевірка продуктивності та впливу виконання Python-коду на основний потік Godot. Особливу увагу приділено ситуаціям, коли виконання коду могло призвести до тимчасового зависання інтерфейсу, тому тестувалися методи асинхронного запуску, обмеження часу виконання та обробка некоректних сценаріїв, що дозволило виявити потенційні вузькі місця та оптимізувати роботу розширення.

Окремий етап містив тестування стійкості системи до помилкових або шкідливих фрагментів коду. Перевірялися крайні випадки: нескінченні цикли, виклики недоступних бібліотек, операції з високими ресурсними витратами. Результати дозволили визначити набір захисних механізмів, які забезпечують стабільність середовища та запобігають критичним зупинкам роботи програми.

Вбудування інтерпретатора Python у середовище Godot забезпечило можливість виконання користувацького коду в інтерактивному режимі та реалізацію функціоналу для навчання мови програмування. Створені механізми виконання, обробки результатів та відображення інформації у користувацькому інтерфейсі дозволяють забезпечити коректну взаємодію між користувачем і середовищем, а також стабільну роботу програми при різних сценаріях використання. Така інтеграція створює надійну основу для подальшого розвитку додатку та підвищення функціональної складності.

Висновок до другого розділу

Проведене дослідження структури розробки кросплатформенного додатка для вивчення мови програмування Python дозволило системно обґрунтувати технологічні та методичні засади створення ефективного інтерактивного навчального середовища. Комплексний аналіз процесу проєктування адаптивного інтерфейсу, інтеграції з серверними сервісами та реалізації вбудованого інтерпретатора продемонстрував доцільність обраних

технічних рішень і їхній вплив на якість користувацької взаємодії та навчальну результативність.

У межах підрозділу 2.1 доведено, що адаптивний дизайн є ключовою передумовою універсальності та доступності освітнього програмного продукту. Практична реалізація адаптивних макетів, використання контейнерних систем Godot, правильне застосування анкорів і параметрів масштабування забезпечили стабільне відображення інтерфейсу на різних розмірах і орієнтаціях дисплеїв. Результати тестування підтвердили, що інтерфейс зберігає логіку навігації, пропорційність елементів і читабельність текстових полів, що є визначальними чинниками для навчальних додатків, орієнтованих на різні вікові групи та технічні можливості користувачів.

У підрозділі 2.2 обґрунтовано значення серверної інтеграції з використанням хмарних сервісів Firebase, яка забезпечує збереження прогресу, авторизацію, захист даних і можливість синхронізації результатів між пристроями. Аналіз архітектури взаємодії між клієнтською частиною Godot і сервером показав, що застосування REST-запитів, структурованих баз даних і вбудованих інструментів автентифікації формує надійну інфраструктуру для реалізації персоналізованих навчальних траєкторій. Такий підхід забезпечує безперервність навчального процесу, легкість масштабування та можливість подальшого розширення функціональності додатка.

У підрозділі 2.3 особливу увагу приділено технічним аспектам упровадження вбудованого інтерпретатора Python як ядра інтерактивного навчального середовища. Використання механізму GDExtension стало основою для інтеграції модулів виконання коду в межах Godot, що дозволило забезпечити безпечне середовище запуску, підтримку базових конструкцій Python, оброблення помилок і виведення результатів у режимі реального часу. Описані рішення підтверджують, що така інтеграція суттєво підсилює педагогічну цінність додатка, оскільки створює умови для негайного

застосування теоретичних знань, формування алгоритмічного мислення та оперативного отримання зворотного зв’язку.

Узагальнюючи результати другого розділу, варто наголосити, що створення кросплатформенного навчального додатка ґрунтується на взаємодоповнювальних технологічних компонентах, серед яких адаптивний дизайн, хмарна інфраструктура та вбудоване середовище виконання коду відіграють визначальні ролі. Їхнє поєднання дозволяє забезпечити високу доступність, гнучкість і масштабованість програмного рішення, а також створює комфортні умови для формування практичних навичок програмування. Запропонована структура доводить, що застосування сучасних технологій у розробленні освітнього програмного забезпечення сприяє підвищенню його ефективності й повністю відповідає актуальним вимогам цифрової трансформації освітнього простору.

ЗАГАЛЬНІ ВИСНОВКИ

Результати підтверджують актуальність розроблення кросплатформенного додатка для вивчення мови програмування Python у освітньому середовищі, що характеризується зростанням потреби у цифрових інструментах навчання, мобільності користувачів та важливості формування алгоритмічного мислення здобувачів освіти. Аналіз теоретичних і прикладних аспектів створення освітніх програмних продуктів засвідчив, що поєднання інтерактивності, адаптивності та доступності є ключовими умовами підвищення ефективності засвоєння матеріалу в галузі програмування.

У першому розділі було присвячено ґрунтовному теоретичному аналізу засад створення кросплатформенних освітніх застосунків та педагогічних особливостей навчання програмуванню мовою Python. У межах першого підрозділу здійснено систематизацію наукових підходів до визначення кросплатформенності як властивості програмного забезпечення функціонувати на різних операційних системах без модифікації кодової бази. Було з’ясовано, що кросплатформені технології забезпечують універсальність доступу до освітнього контенту, знижують витрати на супровід програмного продукту та створюють можливість його використання здобувачами освіти із різними технічними можливостями. Також виділено переваги застосування рушія Godot як гнучкого інструмента, що підтримує логіку побудови навчальних додатків, інтерактивних модулів і візуалізацій.

Подальший теоретичний аналіз було зосереджено на педагогічній і технічній природі мови Python. Доведено, що Python є однією з найбільш придатних мов для навчання через її синтаксичну простоту, читабельність, широкий спектр освітніх матеріалів, а також компактність базових конструкцій, що сприяє швидкому формуванню ключових програмувальних компетентностей. Проаналізовано роботи зарубіжних і українських

дослідників, які підкреслюють ефективність Python як інструмента первинної професійної підготовки.

У першому розділі також систематизовано сучасні підходи до створення навчальних програмних продуктів, зокрема структурованих курсів, інтерфейсних модулів, інтерактивних блоків і систем зворотного зв’язку. Особливу увагу приділено питанням використання адаптивного дизайну, мікросервісної архітектури та інтегрованих інтерпретаторів у навчальних середовищах. Теоретичний аналіз довів, що ефективний навчальний додаток повинен поєднувати інтерактивність, доступність, візуальну зрозумілість, можливість негайного застосування знань і диференціацію навчальних завдань. Таким чином, перший розділ сформував методичну й наукову основу для подальших етапів проєктування програмного продукту.

Другий розділ присвячено практичним аспектам створення кросплатформенного додатка та розкриттю архітектурних і технічних рішень, що сформували основу розробленого програмного продукту. На початковому етапі було проведено моделювання структури додатка, визначено ключові компоненти інтерфейсу, типи екранів, логіку навігації, структуру навчальних модулів і сценарії користувацької взаємодії. Було доведено доцільність застосування адаптивного дизайну, який забезпечує коректне відображення елементів на різних типах пристроїв – від мобільних телефонів до настільних систем.

Особливо важливим напрямом стала реалізація підтримки серверної інфраструктури. У роботі детально описано механізми взаємодії із хмарною платформою Firebase, що забезпечує авторизацію, збереження прогресу та підтримку персоналізованих навчальних траєкторій. Було доведено, що використання хмарних сервісів істотно підвищує надійність і масштабованість навчального додатка, а також створює умови для подальшого розширення функціональності, включно з впровадженням

рейтингових систем, рекомендаційних моделей та інтеграції зі сторонніми освітніми платформами.

Найбільш суттєвим елементом другого розділу стало впровадження вбудованого інтерпретатора Python за допомогою GDExtension. Описано технічні складнощі інтеграції інтерпретатора у середовище Godot, налаштування обробки винятків, передавання даних між модулями, забезпечення безпечного виконання коду та реалізацію функціоналу виведення результатів. Доведено, що реалізація інтерактивного середовища виконання Python-коду істотно підвищує педагогічну ефективність додатка, оскільки забезпечує можливість негайного застосування теоретичних знань, отримання миттєвого зворотного зв’язку та розвиток рефлексивних умінь.

Проведене тестування програмного продукту засвідчило стабільність роботи системи, коректність виконання модулів, відповідність інтерфейсу вимогам доступності та ефективність реалізованих навчальних сценаріїв. Додаток продемонстрував здатність забезпечувати послідовне опанування навчальних тем, підтримувати різні рівні складності та сприяти формуванню навичок аналізу, програмування й усунення помилок.

Загалом отримані результати доводять, що поєднання адаптивного дизайну, серверної інфраструктури та інтерактивного програмного середовища створює комплексний освітній інструмент, здатний ефективно підтримувати процес навчання основам програмування. Створений додаток забезпечує доступність навчального контенту незалежно від платформи, формує умови для практичного опрацювання теоретичних знань та сприяє підвищенню мотивації здобувачів освіти до засвоєння Python.

Особливу увагу приділено педагогічній доцільності використання створеного програмного продукту. Вбудована система виконання коду, структуровані навчальні модулі та можливість роботи в інтерактивному форматі забезпечують розвиток практичних навичок, уміння аналізувати результати роботи програм та знаходити помилки, що є необхідним компонентом професійної підготовки у сфері інформаційних технологій.

Кросплатформенність створює додаткову перевагу для освітніх закладів, де використовується різнорідне технічне оснащення. Усі завдання, визначені темою кваліфікаційної роботи, виконано: досліджено інструменти та технології для створення кросплатформенних додатків; здійснено тестування та оцінювання ефективності розробленого рішення; проведено аналіз теоретичних джерел та сучасних підходів до навчання програмуванню; реалізовано навчальний додаток з інтегрованими модулями виконання Python-коду; спроєктовано архітектуру програмного продукту. Досягнуті результати узгоджуються з метою роботи та підтверджують правильність обраної методології.

Отже, розроблений кросплатформенний додаток може бути рекомендований для використання у формальному та неформальному навчанні програмуванню, у діяльності учителів інформатики, під час самостійної підготовки здобувачів освіти та в системі підвищення кваліфікації педагогічних працівників. Представлена програмна розробка є прикладом ефективного поєднання педагогічних засад і сучасних цифрових технологій, що визначає перспективи її подальшого вдосконалення та масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Базюк Р. Л. Розробка відеогри на базі рушія Godot з використанням мови програмування C# : робота на здобуття кваліфікаційного ступеня бакалавра: спец. 122 Комп'ютерні науки. наук. кер. Л. П. Матійчук. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя. 2025. 57 с.

2. Башуцька О. С., Шакуров Є. О., Шуста В. С. Практичні аспекти впровадження Python у навчанні цифрових дисциплін. Академічні візії (48). 2025. С. 1-12. URL: https://academy-vision.org/index.php/av/article/view/2321

3. Бойко Л. С. Переваги використання flutter у процесі навчання як засобу для заохочення студентів до технічної творчості. In The 7th International scientific and practical conference «Sociological and psychological models of youth communication». Copenhagen, Denmark. International Science Group. 2025. 250 p. (С. 45-46).

4. Васильєв О. Програмування мовою Python. Навчальна книга – Богдан. 2019. 504 с.

5. Довгопол С. О. Аналіз досвіду навчання сучасним мовам програмування у закладах вищої освіти України. Т. 1, № 2. 2018. С. 49-59.

6. Історія Python. E-lab. Блог інженера. Все про електроніку, техніку, IT. 2024. URL: https://e-lab.com.ua/python/istoriia-python.html (дата звернення: 09.05.25)

7. Коцовський В. М. Крос-платформне програмування. Лекції. ЗФН. Електронний репозитарій ДВНЗ «УжНУ». 2017. 12 с. URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/21365/1/%D0%9A%D1%80%D0%BE%D1%81-%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BD%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.%20%D0

%9B%D0%B5%D0%BA%D1%86%D1%96%D1%97.%20%D0%97%D0%A4%D0%9D.pdf

8. Кросплатформне програмування. Факультет інформаційних технологій. Кафедра програмних систем і технологій. Київський національний університет імені Т. Шевченка. URL: https://fit.knu.ua/archives/6186

9. Скорук Д. А., Глинчук Л. Я. Особливості розробки фреймворку для тестування ui частини вебдодатків. *Інформаційні технології і автоматизація*. Матеріали XVII міжнародної науково-практичної конференції. Одеса, Видавництво ОНТУ С. 433-435.

10. Технології розробки кросплатформених веб-додатків. Конспект лекцій. І. В. Пономарьов. Дніпро: ДНУ. 2023. 113 с.

11. Ткаченко О. А., Болячевець Я. Ю. Деякі аспекти розробки та використання сучасних веб-додатків. Інформаційні технології. Водний транспорт: збірник наукових праць. Випуск 2(38). 2023. С. 323-335.

12. Товсточуб І. С., Зінченко О. В. Генерація графічних об’єктів в ігровому рушії Godot за методом паралельних обчислень. Кібербезпека: освіта, наука, техніка. № 3 (27). 2025. С. 479-488.

13. Третяк Д. Мова програмування python як інструмент у науковопедагогічній діяльності. Інформаційні технології і засоби навчання. Том 107, №3. 2025. С. 168-181.

14. Усе про мову програмування Python. SPEKA. URL: https://speka.media/n15469-v7yl83 (дата звернення: 05.05.25)

15. Чичкарьов Є. А., Зінченко О. В., Єльченко С. В. Прикладне програмування на Python. Ч. 1. Основи програмування на Python. Навчальний посібник. Київ: ДУТ. 2022. 160 с.

16. Шаренко А. І. Порівняльний аналіз розробки користувацьких нтерфейсів з компонентною архітектурою і нативними мовами програмування. *Інформаційні та комп’ютерно-інтегровані технології*. VIII

Міжнародна науково-практична конференція «Мехатронні системи: інновації та інжиніринг». С. 214-215.

17. 7 game-changing benefits of cross platform app development: your 2024 implementation guide. Full scale. URL: https://fullscale.io/blog/what-is-cross-platform-app-development/ (дата звернення: 25.02.25)

18. Barry P. Head first. Python. #PROSystem. 2nd edition. Фабула. 2021. 624 с.

19. Beazley D., Jones K. B. Python Cookbook, 3rd Edition. O'Reilly Media, Inc. 2013. 706 p.

20. Bradfield C. Godot 4 Game Development Projects: Build five cross-platform 2D and 3D games using one of the most powerful open source game engines. Second Edition. 2nd Edition. 2023. 264 p.

21. Cross platform mobile app development – pros and cons. Netguru. URL: https://www.netguru.com/blog/cross-platform-mobile-apps-development (дата звернення 09.03.25)

22. Godot engine vs unity: which one suits you best in 2025. Rocketbrush studio LTD. URL: https://rocketbrush.com/blog/godot-vs-unity (дата звернення: 11.09.25)

23. Godot Engine Website. URL: https://godotengine.org (дата звернення: 03.07.25)

24. Godot Engine. URL: https://docs.godotengine.org (дата звернення: 18.07.25)

25. Holfeld J. On the relevance of the Godot Engine in the indie game development industry. University of Kassel. arXiv. 2024. P. 1-9. URL: https://www.researchgate.net/publication/383116776_On_the_relevance_of_the_Godot_Engine_in_the_indie_game_development_industry

26. Ivashkevych L. Teaching Programming with Python for Linguists: Whys and How-tos. Advanced Linguistics, 2019. С. 4-12.

27. Learn the 5 most important features of Godot engine with us!. Logiscool. URL: https://www.logiscool.com/en/blog/coding-tech/learn-the-5-most-important-features-of-godot-engine-with-us (дата звернення: 10.06.25)

28. Lowood H. Game engines and game history. in history of gamesinternational conference proceedings. Kinephanos. 2014. P. 179-198. URL: https://www.kinephanos.ca/Revue_files/2014-Lowood.pdf

29. Lutz M. Learning Python, Fourth Edition. O'Reilly Media, Inc. 2009. 1213 p.

30. Manzur A. Godot Engine – multi-platform 2D and 3D game engine. GitHub. 2025. URL: https://github.com/godotengine/godot (дата звернення 13.02.25)

31. Matthes E. Python crash course, 3rd edition: a hands-on, project-based introduction to programming. 2023. 552 p.

32. Mimo: Learn to Code. URL: https://getmimo.com/ (дата звернення: 12.05.25)

33. Mobile operating system market share worldwide. Mar 2024 – mar 2025. Statcounter. GlobalStats. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата звернення: 11.04.25)

34. Native vs cross-platform mobile app development. Circleci blog. URL: https://circleci.com/blog/native-vs-cross-platform-mobile-dev/ (дата звернення: 08.03.25)

35. Native, hybrid, or cross-platform apps? Microsoft. URL: https://www.microsoft.com/en-us/power-platform/products/power-apps/topics/app-development/native-vs-cross-platform-apps (дата звернення: 18.04.25)

36. Obvious advantages of cross-platform development. Link up. URL: https://linkupst.com/blog/advantages-of-cross-platform-development (дата звернення: 16.03.25)

37. Official Python Website. URL: https://www.python.org/ (дата звернення: 04.02.25)

38. Project Jupyter. URL: https://jupyter.org/ (дата звернення: 14.05.25)

39. Python he language of today and tomorrow. Python institute (PI). URL: https://pythoninstitute.org/about-python (дата звернення: 12.12.25)

40. Ramalho L. Fluent Python, 2nd Edition. Intermediate to advanced. O'Reilly Media, Inc. 2022. 1014 p.

41. Replit - The collaborative browser-based IDE. URL: https://replit.com/ (дата звернення: 12.05.25)

42. Shokaliuk S. V., Bohunenko Y. Y. Lovianova I. V., Shyshkina M. P. Technologies of distance learning for programming basics on the principles of integrated development of key competences. CTE Workshop Proceedings. Vol. 7: CTE-2019. 2020. P. 548-562.

43. Stack Overflow Developer Survey. URL: https://survey.stackoverflow.co (дата звернення: 15.05.25)

44. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. 1st Edition. No Starch Press. 2019. 504 p.

45. The advantages of cross-platform mobile app development. Eleks. URL: https://eleks.com/types-of-software-development/the-advantages-of-cross-platform-mobile-app-development/ (дата звернення: 22.03.25)

46. The pros and cons of cross-platform mobile app development frameworks. Taazaa. URL: https://www.taazaa.com/cross-platform-mobile-app-development/ (дата звернення: 11.02.25)

47. The six most popular cross-platform app development frameworks. Jet brains. Kotlin multiplatform journal. URL: https://www.jetbrains.com/help/kotlin-multiplatform-dev/cross-platform-frameworks.html (дата звернення: 18.02.25)

48. Thorn A. Game Development with Godot 4. Computer Science. CRC Press. 2025. 376 p. URL: https://www.perlego.com/book/5218349/game-development-with-godot-4-a-complete-introduction-pdf

49. Why Godot is right for you. Godot. URL: https://godotengine.org/features/ (дата звернення: 15.07.25)

50. Williams A. History of digital games: developments in art, designand interaction. CRCPress, Taylor & Francis Group. 2017. P. 152-154.