UNICHECK

Ім'я користувача:
приховано налаштуваннями конфіденційності

ID перевірки:
1015591162

Дата перевірки:
13.06.2023 20:09:46 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
13.06.2023 20:10:20 EEST

ID користувача:
100011372

Назва документа: **КН320 Михайлович**

Кількість сторінок: **32** Кількість слів: **7671** Кількість символів: **60866** Розмір файлу: **113.29 KB** ID файлу: **1015240367**

1.94%
Схожість

Найбільша схожість: **0.38%** з джерелом з Бібліотеки (ID файлу: **1015192133**)

Пошук збігів з Інтернетом не проводився

1.94% Джерела з Бібліотеки | **85** ..... Сторінка 34

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

1 ОГЛЯД СУЧАСНИХ ЗАСОБІВ РОЗРОБКИ МОБІЛЬНИХ ДОДАТКІВ

1.1 Класифікація та характеристика сучасних засобів розробки мобільних додатків

Зростання мобільних пристроїв призвело до різкого зростання попиту на мобільні додатки. Отже, ринок розробки мобільних додатків швидко розвивався, пропонуючи широкий спектр інструментів і фреймворків для задоволення цієї потреби. У цьому розділі ми дослідимо класифікацію та характеристики сучасних інструментів розробки мобільних додатків, зосередившись на трьох основних категоріях: нативні, гібридні та веб-фреймворки.

Рідні засоби розробки

Власні інструменти розробки дозволяють розробникам створювати програми, які спеціально адаптовані для однієї платформи, наприклад Android або iOS. Ці інструменти надають доступ до API платформи та апаратних функцій , забезпечуючи високий рівень налаштування та оптимізації продуктивності .

характеристики

Спеціально для платформи: рідні програми розробляються з використанням мов і фреймворків, специфічних для платформи, таких як Java або Kotlin для Android і Swift або Objective-C для iOS.

Висока продуктивність: оскільки рідні програми написані явно для цільової платформи, їх можна оптимізувати для досягнення високої продуктивності шляхом використання специфічних для платформи функцій і апаратних можливостей.

Доступ до власних API: власні інструменти розробки надають широкий доступ до платформних API, дозволяючи розробникам створювати програми з розширеними функціями, такими як геолокація, доступ до камери та push-повідомлення .

Взаємодія з користувачем (UX): рідні програми можуть забезпечити багатий і безперебійний досвід роботи з користувачем, дотримуючись інструкцій щодо

дизайну для певної платформи та пропонуючи власні компоненти інтерфейсу користувача .

Приклади

AndroidStudio : офіційна IDE для розробки Android, AndroidStudio заснована на IntelliJ IDEA та пропонує повний набір інструментів для створення програм Android , включаючи потужний редактор коду , редактор візуального макета та вбудований емулятор.

Xcode: офіційна IDE від Apple для розробки iOS і macOS, Xcode надає широкий спектр інструментів і функцій для створення програм, включаючи потужний редактор коду, InterfaceBuilder і інтегровані симулятори.

Інструменти гібридної розробки

Інструменти гібридної розробки дозволяють розробникам створювати кросплатформні мобільні програми за допомогою єдиної кодової бази . Ці програми можуть працювати на кількох платформах, таких як Android, iOS і Windows, пропонуючи економічно ефективний підхід до мобільної розробки, що економить час.

характеристики

Кросплатформенність: гібридні програми розробляються з використанням веб-технологій, таких як HTML , CSS і JavaScript , які можна виконувати на кількох платформах з мінімальними налаштуваннями для конкретної платформи.

Єдина кодова база: інструменти гібридної розробки дозволяють розробникам підтримувати єдину кодову базу, скорочуючи час і зусилля на розробку, забезпечуючи узгодженість між платформами.

Доступ до власних функцій. Більшість гібридних фреймворків надають доступ до власних функцій через плагіни або розширення, що дозволяє розробникам включати функціональні можливості, специфічні для платформи, хоча й з деякими обмеженнями порівняно з власною розробкою.

Компроміси продуктивності: гібридні програми можуть не запропонувати такий самий рівень продуктивності, як рідні програми, оскільки вони працюють у WebView або подібному контейнері, що може спричинити додаткові витрати.

Приклади

ReactNative : розроблений Facebook , ReactNative дозволяє розробникам створювати міжплатформні програми за допомогою JavaScript і Reactframework . Він забезпечує нативний досвід, відтворюючи власні компоненти інтерфейсу користувача та дозволяючи доступ до специфічних для платформи API.

Flutter : створений Google , Flutter — це набір інструментів інтерфейсу користувача для створення власно скомпільованих програм за допомогою мови програмування Dart . Він пропонує багатий набір віджетів та інструментів для створення адаптивних і візуально привабливих програм.

Засоби веб-розробки

Інструменти веб-розробки дозволяють розробникам створювати мобільні програми, які працюють у мобільному браузері або компоненті WebView. Ці програми, по суті, адаптивні веб-програми , доступ до яких можна отримати з різних пристроїв, включаючи смартфони та планшети.

характеристики

Незалежність від платформи: веб-програми створено з використанням стандартних веб-технологій , таких як HTML, CSS і JavaScript, і до них можна отримати доступ на різних платформах і пристроях.

Просте обслуговування: оскільки веб-програми покладаються на єдину кодову базу, їх легше підтримувати й оновлювати порівняно з рідними або гібридними програмами.

Обмежений доступ до власних функцій: веб-програми мають обмежений доступ до власних функцій і можуть не надавати такий самий рівень функціональності чи взаємодії з користувачем, як рідні або гібридні програми.

Зауваження щодо продуктивності: веб-програми можуть страждати від проблем із продуктивністю , особливо при повільніших або ненадійних мережевих з’єднаннях, оскільки вони покладаються на веб-технології та віддалені ресурси.

Приклади

ApacheCordova (раніше PhoneGap): Apache Cordova дозволяє розробникам створювати веб-додатки за допомогою HTML, CSS і JavaScript і пакувати їх як рідні додатки за допомогою компонентів WebView для певної платформи.

Прогресивні веб-програми (PWA): PWA — це веб-програми , які можна встановити на пристрої користувача та пропонують автономні можливості , push-сповіщення та інші нативні функції за допомогою сучасних веб-API .

Висновок

Підсумовуючи, інструменти розробки мобільних додатків можна загалом класифікувати на власні, гібридні та веб-фреймворки. Кожна категорія має власний набір характеристик, переваг і недоліків, а вибір правильного інструменту залежить від таких факторів, як вимоги до проекту, бюджет і бажаний досвід користувача. Завдяки безперервному розвитку технологій мобільної розробки розробникам надається різноманітний набір інструментів і фреймворків для створення інноваційних і привабливих мобільних додатків.

Мобільні програми стали невід'ємною частиною сучасного життя , мільйони людей користуються ними щодня. У результаті розробка мобільних додатків стала бурхливою галуззю. Процес розробки мобільних додатків став простішим і доступнішим за допомогою сучасних засобів розробки мобільних додатків. У цьому есе ми обговоримо класифікацію та характеристики сучасних засобів розробки мобільних додатків. Класифікація засобів розробки мобільних додатків Засоби розробки мобільних додатків можна класифікувати на три основні категорії залежно від мови програмування, яка використовується для розробки: Власні інструменти розробки: ці інструменти використовуються для розробки мобільних додатків з використанням мов програмування для певної платформи, таких як Java для Android , Swift для iOS і C# для Windows. Власні інструменти розробки пропонують чудову продуктивність, безперебійну взаємодію з користувачем і доступ до специфічних для платформи функцій. Однак вони вимагають від розробників певних знань і досвіду щодо платформи . Інструменти гібридної розробки: ці інструменти використовуються для розробки мобільних програм із використанням веб-технологій, таких як HTML5 , CSS і JavaScript .

Інструменти гібридної розробки дозволяють розробникам створювати єдину кодову базу , яку можна розгорнути на кількох платформах. Вони пропонують швидший цикл розробки, простіше обслуговування та доступ до таких функцій пристрою, як камера, акселерометр і GPS. Однак вони можуть не забезпечувати такої ж продуктивності та взаємодії з користувачем, як рідні інструменти розробки. Інструменти міжплатформної розробки : ці інструменти використовуються для розробки мобільних додатків за допомогою однієї кодової бази, яку можна розгортати на кількох платформах. Інструменти кросплатформенної розробки дозволяють розробникам використовувати різні мови програмування, такі як C#, JavaScript і Java. Вони пропонують швидший цикл розробки, спрощене обслуговування та доступ до специфічних для платформи функцій. Однак вони можуть не забезпечувати такої ж продуктивності та взаємодії з користувачем, як рідні інструменти розробки. Характеристика засобів розробки мобільних додатків Дизайн інтерфейсу користувача : сучасні засоби розробки мобільних додатків пропонують різноманітні варіанти дизайну інтерфейсу користувача, такі як інтерфейси перетягування, візуальні редактори та готові шаблони. Ці функції дозволяють розробникам створювати візуально привабливі та зручні мобільні програми. Інтеграція зі сторонніми службами: мобільні програми часто вимагають інтеграції зі сторонніми службами, такими як платформи соціальних мереж, платіжні шлюзи та служби хмарного зберігання даних . Сучасні засоби розробки мобільних додатків пропонують просту інтеграцію з такими службами, дозволяючи розробникам створювати багатофункціональні мобільні додатки. Тестування та налагодження: Тестування та налагодження є важливими аспектами розробки мобільних додатків. Сучасні засоби розробки мобільних додатків пропонують такі функції, як автоматичне тестування, налагодження в реальному часі та звітування про помилки, що полегшує розробникам виявлення та усунення проблем. Співпраця та контроль версій: розробка мобільних додатків часто передбачає співпрацю кількох розробників. Сучасні засоби розробки мобільних додатків пропонують такі функції співпраці, як контроль версій, спільний доступ до коду та командне

спілкування, що полегшує спільну роботу розробників над одним проектом. Висновок Інструменти розробки мобільних додатків пройшли довгий шлях, і сучасні інструменти пропонують різноманітні функції та функції, які роблять процес розробки простішим, швидшим і доступнішим. Розробники можуть вибирати з різноманітних інструментів на основі своїх вимог і переваг, а класифікація та характеристики цих інструментів можуть допомогти розробникам приймати зважені рішення. За допомогою сучасних засобів розробки мобільних додатків розробники можуть створювати високоякісні мобільні додатки, які відповідають потребам мільйонів користувачів у всьому світі. Розкажи мені більше. Звичайно, ось деякі додаткові відомості про класифікацію та характеристики сучасних засобів розробки мобільних додатків: Власні засоби розробки: Власні засоби розробки ідеально підходять для розробки високопродуктивних мобільних додатків, які забезпечують безперебійну роботу користувача . Ці інструменти пропонують доступ до специфічних для платформи функцій, таких як камера, місцезнаходження та push-сповіщення , що дозволяє розробникам створювати багатофункціональні програми. Проте власні інструменти розробки вимагають певних знань і досвіду щодо платформи, що може стати перешкодою для деяких розробників. Інструменти гібридної розробки: інструменти гібридної розробки ідеально підходять для розробки мобільних додатків, які потрібно швидко й ефективно розгортати на кількох платформах. Ці інструменти пропонують швидший цикл розробки, легше обслуговування та доступ до таких функцій пристрою, як камера, акселерометр і GPS. Однак гібридні інструменти розробки можуть не забезпечувати такий самий рівень продуктивності та взаємодії з користувачем, як власні засоби розробки. Інструменти кросплатформенної розробки: інструменти кросплатформенної розробки ідеально підходять для розробки мобільних додатків, які потрібно розгортати на кількох платформах, зберігаючи узгоджену взаємодію з користувачем. Ці інструменти пропонують швидший цикл розробки, спрощене обслуговування та доступ до специфічних для платформи функцій. Однак кросплатформні інструменти розробки можуть не забезпечувати такий самий

рівень продуктивності та взаємодії з користувачем, як рідні інструменти розробки. Дизайн інтерфейсу користувача : сучасні засоби розробки мобільних додатків пропонують різноманітні варіанти дизайну інтерфейсу користувача , включаючи інтерфейси перетягування, візуальні редактори та готові шаблони. Ці функції дозволяють розробникам створювати візуально привабливі та зручні мобільні програми, які відповідають потребам цільової аудиторії. Інтеграція зі сторонніми службами: мобільні програми часто вимагають інтеграції зі сторонніми службами, такими як платформи соціальних мереж, платіжні шлюзи та служби хмарного зберігання даних . Сучасні засоби розробки мобільних додатків пропонують просту інтеграцію з такими службами, дозволяючи розробникам створювати багатофункціональні мобільні додатки, які відповідають потребам користувачів. Тестування та налагодження. Тестування та налагодження є важливими аспектами розробки мобільних додатків . Сучасні засоби розробки мобільних додатків пропонують такі функції, як автоматичне тестування , налагодження в реальному часі та звітування про помилки , що полегшує розробникам виявлення та усунення проблем. Співпраця та контроль версій : розробка мобільних додатків часто передбачає співпрацю кількох розробників. Сучасні засоби розробки мобільних додатків пропонують такі функції співпраці , як контроль версій, спільний доступ до коду та командне спілкування, що полегшує спільну роботу розробників над одним проектом. Підсумовуючи, сучасні засоби розробки мобільних додатків пропонують різноманітні функції та функції, які роблять процес розробки простішим, швидшим і доступнішим. Розробники можуть вибирати з різноманітних інструментів на основі своїх вимог і переваг, а класифікація та характеристики цих інструментів можуть допомогти розробникам приймати зважені рішення. За допомогою сучасних засобів розробки мобільних додатків розробники можуть створювати високоякісні мобільні додатки, які відповідають потребам мільйонів користувачів у всьому світі.

1.2 Огляд мов програмування для розробки мобільних додатків

Для розробки мобільних додатків можна використовувати різні мови програмування. Основні мови, які виконуються для розробки мобільних додатків, це Java, Kotlin для Android, Swift і Objective-C для iOS, C# для Windows Phone. Також для розробки мобільних додатків можна використовувати мови програмування, які вибирають для веб-розробки, такі як HTML, CSS та JavaScript.

Мові програмування для розробки мобільних додатків можуть бути класифіковані за допомогою двох підходів:

Мова програмна платформа: Цей підхід використовує мови програмування, які спеціально розроблені для певної платформи, наприклад, Java і Kotlin для Android, Swift і Objective-C для iOS, і C# для Windows Phone. Ці мови програмування найбільш оптимізовані для розробок під платформу, що забезпечують високу продуктивність та взаємодію зі специфічними характеристиками платформи.

Мова програмування загального призначення: цей підхід використовує мови програмування, які можна використовувати для розробки на різних платформах, наприклад, HTML, CSS і JavaScript. Ці мови програмування можуть бути використані для розробки гібридних та крос-платформових додатків.

Мові програмування для розробки мобільних додатків мають ряд особливостей, які можуть розробникам створювати високоякісні додатки для різних платформ. Основні особливості мов програмування для розробки мобільних додатків включають:

Підтримка функцій платформи: Мові програмування для розробки мобільних додатків повинні мати доступ до певних функцій платформи, таких як камера, GPS та інші. Це дозволяє розробникам створювати додатки з більшою функціональністю та можливостями.

Продуктивність: Додатки повинні бути швидкими та ефективними, оскільки мови програмування для розробки мобільних додатків повинні бути оптимізовані для роботи на мобільних пристроях та забезпечувати високу продуктивність.

Безпека: Мобільні додатки мають доступ до конфіденційної інформації користувачів, тому мови програмування для розробки мобільних додатків повинн

Java: Java є популярною мовою програмування для розробки додатків Android . Він пропонує надійне середовище розробки, ефективне керування пам'яттю та легкий доступ до основних бібліотек Android . Java також є універсальною мовою, яку можна використовувати для розробки настільних і веб-додатків .

Kotlin: Kotlin — відносно нова мова програмування, яка набирає популярності серед розробників Android. Це офіційно підтримувана мова для розробки Android і пропонує ряд функцій, які покращують продуктивність, наприклад нульовий захист , функції розширення та класи даних. Kotlin також сумісний з Java, що означає, що існуючий код Java можна легко інтегрувати в проекти Kotlin.

Swift: Swift — це мова програмування, розроблена Apple для iOS, macOS, watchOS і tvOS . Він пропонує сучасний синтаксис, покращену продуктивність і просте керування пам'яттю . Swift також зворотно сумісний з Objective-C, що означає, що існуючий код Objective-C можна легко інтегрувати в проекти Swift.

Objective-C: Objective-C — це мова програмування, яка вже багато років використовується для розробки iOS. Він пропонує динамічне середовище виконання , об'єктно- орієнтоване програмування та легку інтеграцію з кодом C і C++. Objective-C все ще широко використовується в розробці iOS, але його поступово замінює Swift.

C#: C# — це мова програмування, розроблена Microsoft для розробки Windows Phone . Він пропонує сучасний синтаксис , просте керування пам'яттю та доступ до середовища .NET. C# також є універсальною мовою, яку можна використовувати для розробки настільних додатків, веб-додатків та ігор.

HTML , CSS і JavaScript : HTML, CSS і JavaScript — це веб-технології , які можна використовувати для розробки гібридних і кросплатформних мобільних програм. HTML забезпечує структуру програми, CSS використовується для стилізації, а JavaScript використовується для додавання інтерактивності та

функціональності. Перевага використання веб-технологій для розробки мобільних додатків полягає в тому, що ту саму кодову базу можна використовувати на кількох платформах.

ReactNative: React Native — це фреймворк із відкритим кодом , розроблений Facebook для створення нативних мобільних додатків за допомогою JavaScript і React. Це дозволяє розробникам створювати програми як для платформ iOS, так і для Android , використовуючи єдину кодову базу. ReactNative також надає доступ до нативних API і пропонує швидкий цикл розробки.

Flutter: Flutter — це фреймворк із відкритим кодом, розроблений Google для створення власно скомпільованих мобільних додатків за допомогою мови програмування Dart . Це дозволяє розробникам створювати програми як для платформ iOS, так і для Android , використовуючи єдину кодову базу. Flutter також пропонує швидкий цикл розробки, гаряче перезавантаження та багатий набір настроюванихвіджетів .

Xamarin : Xamarin — це міжплатформна платформа для створення мобільних програм за допомогою C# і .NET. Це дозволяє розробникам створювати програми для платформ iOS, Android і Windows Phone , використовуючи єдину кодову базу. Xamarin також надає доступ до власних API і пропонує швидкий цикл розробки.

Ionic : Ionic — це фреймворк із відкритим вихідним кодом для створення гібридних мобільних програм із використанням таких веб-технологій , як HTML , CSS і JavaScript. Це дозволяє розробникам створювати програми для платформ iOS, Android і Windows Phone , використовуючи єдину кодову базу. Ionic також надає багатий набір компонентів інтерфейсу користувача та велику бібліотеку плагінів для інтеграції з рідними API.

PhoneGap: PhoneGap — це платформа з відкритим кодом для створення гібридних мобільних додатків із використанням таких веб-технологій, як HTML, CSS і JavaScript. Це дозволяє розробникам створювати програми для платформ iOS, Android і Windows Phone, використовуючи єдину кодову базу. PhoneGap також забезпечує швидкий цикл розробки та доступ до рідних API.

Розробка бекенда: окрім розробки програм для мобільних пристроїв, розробникам також необхідно розглянути розробку бекенда, яка передбачає розробку серверної логіки , API та баз даних для підтримки програми. До популярних серверних технологій належать Node.js, RubyonRails , Django та Flask.

Хмарні служби : розробники також можуть використовувати такі хмарні служби , як AmazonWebServices (AWS), Microsoft Azure і GoogleCloudPlatform , щоб розмістити свою серверну інфраструктуру , керувати автентифікацією користувачів і зберігати дані програм.

Дизайн інтерфейсу користувача. Дизайн інтерфейсу користувача (UI) є важливим аспектом розробки мобільних додатків. Добре розроблений інтерфейс користувача може покращити залучення та утримання користувачів. Розробники можуть використовувати інструменти дизайну, такі як Sketch, Adobe XD або Figma, для створення макетів інтерфейсу користувача та прототипів.

Тестування та налагодження. Тестування та налагодження є важливими аспектами розробки мобільних додатків . Розробникам потрібно тестувати свої програми на різних пристроях, операційних системах і мережевих умовах, щоб переконатися, що вони працюють належним чином. Вони можуть використовувати інструменти тестування, такі як Appium , Espresso або XCUITest , щоб автоматизувати тестування та налагодження.

Надсилання в AppStore : після того, як програму розроблено та протестовано, її потрібно надіслати в AppStore для розповсюдження. Розробники повинні дотримуватися вказівок магазину програм щодо надсилання програм і переконатися, що їхні програми відповідають вимогам магазину програм.

Оптимізація продуктивності : продуктивність мобільного додатка має вирішальне значення для взаємодії з користувачем. Розробникам потрібно оптимізувати свої програми для швидшого завантаження , більш плавної анімації та зменшення споживання заряду батареї. Вони можуть використовувати такі інструменти, як AndroidProfiler, XcodeInstruments або ReactNative PerformanceMonitor, щоб виявити та виправити проблеми з продуктивністю.

Безпека. Безпека мобільних програм також важлива для захисту даних користувачів, запобігання несанкціонованому доступу та забезпечення цілісності програми. Розробники повинні впровадити такі заходи безпеки, як шифрування даних, безпечна автентифікація та безпечне зберігання даних. Вони також можуть використовувати інструменти тестування безпеки , такі як OWASP ZAP або BurpSuite, щоб перевірити вразливість свого додатка.

Аналітика: аналітика може надати цінну інформацію про те, як користувачі взаємодіють із додатком, які функції вони використовують найчастіше та які сфери потребують покращення. Розробники можуть використовувати інструменти аналітики, такі як GoogleAnalytics, FirebaseAnalytics або Mixpanel, щоб відстежувати поведінку користувачів і продуктивність програми.

Технічне обслуговування та оновлення: після розгортання програми розробники повинні забезпечити регулярне технічне обслуговування та оновлення, щоб гарантувати, що програма продовжує працювати належним чином і відповідати потребам користувачів. Їм потрібно виправляти помилки, додавати нові функції та оптимізувати продуктивність на основі відгуків користувачів і аналітики.

Монетизація: монетизація є важливим аспектом розробки мобільних додатків, оскільки вона дозволяє розробникам отримувати дохід від своїх додатків. Існують різні способи монетизації мобільних програм , зокрема покупки в програмі, підписки, реклама та спонсорство. Розробники повинні вибрати стратегію монетизації , яка найкраще підходить для їх програми та користувачів.

Спеціальні можливості: доступність є важливим фактором у розробці мобільних програм , оскільки вона гарантує, що програмою можуть користуватися люди з обмеженими можливостями. Розробники повинні проектувати та розробляти свої програми, щоб вони були доступними для людей із вадами зору, слуху чи моторики. Вони можуть використовувати такі інструменти доступності , як VoiceOver, TalkBack або SwitchControl , щоб перевірити доступність своєї програми.

Інтернаціоналізація: Інтернаціоналізація – це процес проектування та розробки додатків, які можуть використовувати люди з різних країн і культур. Під час розробки програм розробникам необхідно враховувати такі фактори, як мова, валюта та культурні відмінності. Вони можуть використовувати інструменти локалізації, такі як AndroidStudio або Xcode , щоб перекладати вміст своєї програми різними мовами.

Відгуки користувачів: відгуки користувачів є критично важливими для успіху мобільних програм. Розробникам потрібно збирати відгуки користувачів і використовувати їх для покращення функціональності, зручності використання та продуктивності свого додатка. Вони можуть використовувати інструменти зворотного зв'язку, такі як опитування в додатку , рейтинги додатків або соціальні мережі, щоб зібрати відгуки користувачів.

Гнучка розробка: гнучка розробка — це популярний підхід до розробки мобільних додатків, який наголошує на співпраці, гнучкості та постійному вдосконаленні. Це передбачає розбиття процесу розробки на невеликі ітераційні цикли та регулярне тестування та інтеграцію нових функцій. Гнучка розробка дозволяє розробникам швидко реагувати на зміни потреб користувачів і ринкових умов.

Управління проектами: ефективне управління проектами має вирішальне значення для успіху розробки мобільних додатків. Керівники проектів повинні планувати та координувати процес розробки, розподіляти ресурси, відстежувати прогрес і спілкуватися із зацікавленими сторонами. Вони можуть використовувати такі інструменти управління проектами, як Asana, Trello або Jira, щоб керувати своїми проектами.

Командна співпраця: розробка мобільних додатків часто передбачає співпрацю між кількома командами, такими як розробники, дизайнери, тестувальники та менеджери проектів. Ефективна співпраця команди має важливе значення, щоб гарантувати, що всі працюють над тими самими цілями та що додаток буде доставлено вчасно та в межах бюджету. Такі інструменти для

співпраці, як Slack, Microsoft Teams або GoogleWorkspace, можуть допомогти командам ефективно спілкуватися та співпрацювати.

Дизайн, орієнтований на користувача: дизайн, орієнтований на користувача , — це підхід до дизайну, який зосереджується на розумінні та задоволенні потреб користувачів. Це передбачає збір відгуків користувачів, проведення досліджень користувачів і розробку функцій та інтерфейсу програми на основі потреб і вподобань користувачів. Дизайн, орієнтований на користувача, може покращити зручність використання, функціональність і загальний досвід роботи з мобільними програмами .

Штучний інтелект: Штучний інтелект (ШІ) стає все більш важливим у розробці мобільних додатків. AI можна використовувати для покращення взаємодії з користувачем , персоналізації вмісту та автоматизації завдань. Розробники можуть використовувати такі фреймворки ШІ, як TensorFlow, Keras або PyTorch , щоб інтегрувати ШІ у свої програми.

Доповнена реальність: доповнена реальність (AR) — це ще одна нова технологія, яка може покращити функціональність мобільних додатків і покращити взаємодію з користувачем. AR дозволяє розробникам накладати цифровий контент на реальний світ, надаючи користувачам більш захоплюючий та інтерактивний досвід. Фреймворки AR, такі як ARKit , ARCore або Vuforia , можна використовувати для розробки програм із підтримкою AR.

Блокчейн: технологію блокчейн також можна використовувати в розробці мобільних додатків для забезпечення безпечних і прозорих транзакцій і зберігання даних. Блокчейн можна використовувати для розробки програм для управління цифровими активами , безпечного обміну даними та децентралізованих програм. Інфраструктури блокчейну, такі як Ethereum, Hyperledger або Corda , можна використовувати для розробки додатків із підтримкою блокчейну.

Інтернет речей: Інтернет речей (IoT) — ще одна технологічна тенденція, яка впливає на розробку мобільних додатків. IoT дозволяє користувачам підключати та керувати такими пристроями, як розумна побутова техніка, переносні пристрої

та датчики, за допомогою своїх мобільних пристроїв. Розробники можуть використовувати фреймворки IoT , такі як AWS IoT, GoogleCloudIoT або AzureIoT, для розробки мобільних програм із підтримкою IoT.

Конфіденційність: конфіденційність є важливим фактором у розробці програм для мобільних пристроїв, оскільки передбачає захист даних користувачів і забезпечення відповідності програми положенням щодо конфіденційності, таким як Загальний регламент захисту даних (GDPR) або Закон Каліфорнії про конфіденційність споживачів (CCPA). Розробникам необхідно впровадити такі заходи конфіденційності, як шифрування даних, безпечна автентифікація та мінімізація даних.

Гейміфікація: Гейміфікація – це техніка, яка передбачає додавання ігрових елементів до неігрового контексту, наприклад мобільних додатків, для підвищення залученості та мотивації. Елементи гейміфікації можуть включати значки, бали, таблиці лідерів або нагороди. Розробники можуть використовувати гейміфікацію, щоб покращити залучення та утримання користувачів у своїх програмах.

Соціальна інтеграція: соціальна інтеграція передбачає інтеграцію платформ соціальних мереж, таких як Facebook , Twitter або Instagram , у мобільні програми, щоб дозволити користувачам ділитися вмістом, спілкуватися з друзями або входити за допомогою своїх облікових записів у соціальних мережах. Соціальна інтеграція може покращити залучення користувачів і вірусність мобільних додатків.

Глибинні посилання: Глибокі посилання дозволяють користувачам переходити безпосередньо до певної сторінки чи функції в мобільній програмі замість того, щоб відкривати домашню сторінку програми. Розробники можуть використовувати глибокі посилання, щоб покращити взаємодію з користувачем і збільшити використання програми.

Хмарні обчислення: Хмарні обчислення — це надання обчислювальних послуг, таких як сервери, сховище, бази даних і програмне забезпечення, через Інтернет. Хмарні обчислення можна використовувати в розробці мобільних додатків, щоб забезпечити масштабовану та гнучку інфраструктуру, зменшити

витрати та підвищити продуктивність додатків. Такі постачальники хмарних послуг, як AmazonWebServices (AWS), Microsoft Azure або GoogleCloudPlatform (GCP), пропонують різні хмарні обчислювальні послуги, які можна використовувати для розробки мобільних додатків.

Прогресивні веб-програми : Прогресивні веб-програми (PWA) — це веб-програми, які надають користувачам нативний досвід, схожий на програму, зокрема доступ у режимі офлайн, push-повідомлення та доступ до апаратного забезпечення пристрою. PWA можна отримати через веб-браузери та встановити на головних екранах користувачів. Розробники можуть використовувати фреймворки PWA, такі як Ionic, React або Angular, для розробки PWA.

Wearables: Wearables, такі як розумні годинники або фітнес-трекери, стають дедалі популярнішими, і розробники мобільних додатків шукають способи інтегрувати їх у свої додатки. Інтеграція переносних пристроїв може надати користувачам нові та інноваційні способи взаємодії з мобільними додатками .

Голосові помічники: Голосові помічники, такі як AmazonAlexa або GoogleAssistant , також стають все більш поширеними, і розробники мобільних додатків можуть інтегрувати їх у свої додатки, щоб забезпечити вільні руки та більш природний досвід користувача. Голосову інтеграцію можна використовувати для керування програмами, доступу до інформації або у досвід користувача. галузь розробки мобільних додатків.

У тексті розглянуто різні аспекти розробки мобільних додатків, включаючи вибір мови програмування та фреймворку, оптимізацію продуктивності, безпеку, монетизацію, доступність, управління проектами, командну співпрацю, дизайн, інноваційні технології та інші. Враховуючи ці фактори, розробники можуть створювати високоякісні програми, які відповідають потребам користувачів і досягають бізнес-цілей. Для досягнення успіху в індустрії розробки мобільних додатків важливо відстежувати нові технології та тенденції.

2 РОЗРОБКА ДИСТАНЦІЙНОГО КУРСУ З ДИСЦИПЛІНИ "ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

2.1 Розробка структури навчального курсу

У сучасному світі де швидкість змін та розвитку технологій надзвичайно висока, важливо мати можливість навчатися новим навичкам та здобувати нові знання в будь-який час та з будь-якого місця. У зв'язку з цим дистанційне навчання стає все більш популярним. І цей найбільш популярний вид дистанційного навчання є дистанційний курс.

У даній дипломній роботі розглянемо процес розробки дистанційного курсу з дисципліни "Об'єктно-орієнтоване програмування". Для розробки курсу було використано платформу для створення додатків на телефон Android Studio.

Першим етапом розробки курсу було створення його структури. Для цього було визначено загальну мету курсу та розділено її на конкретні теми. Кожна тема містила достатньо матеріалів для навчання, таких як відеоуроки, текстові матеріали, завдання для самостійної роботи та тести для перевірки знань.

Другим етапом було створено відеоуроків та текстових матеріалів для кожної теми курсу. Відеоуроки були створені з використанням програми поняття екрану та звуку, яка дозволяла записувати рухи курсора та голосовий коментар. Текстові матеріали були створені у форматі PDF та містили теоретичні відомості та приклади для кращого розуміння матеріалу.

Третім етапом було створення завдань для самостійної роботи. Завдання містили практичні вправи та завдання з програмування, які допомогли закріпити отримані знання та навички.

Четвертим етапом було створення тестів для перевірки знань. Тести були створені у форматі питання-відповідь та містили питання з теорії та практичні завдання з програмування.

Після створення всіх матеріалів курс було завантажено на платформу і розміщено в доступі для користувачів. Кожен користувач може зареєструватися на курсі та пройти його в будь-який зручний для нього час.

Першим кроком у розробці дистанційного курсу є визначення цілей навчання та відповідна структура курсу. Це включає розбивку курсу на модулі або теми, окреслення навчальних цілей для кожного та визначення найкращих методів навчання для досягнення цих цілей.

Другим кроком є розробка змісту курсу , який включає навчальні матеріали, такі як відео, тексти та завдання. Для курсу об'єктно-орієнтованого програмування відео можна використовувати для пояснення таких понять, як успадкування, поліморфізм та інкапсуляція, тоді як читання може надати додаткову інформацію про синтаксис мови програмування та найкращі практики. Завдання можуть включати вправи з програмування , завдання з усунення помилок і тести.

Третій крок — вибір системи керування навчанням (LMS) або платформи для надання змісту курсу. Існує багато доступних варіантів, включаючи платформи з відкритим кодом, такі як Moodle або Canvas , а також комерційні інструменти, такі як Blackboard або Brightspace . LMS має надавати студентам зручний інтерфейс для доступу до матеріалів курсу та взаємодії з викладачами та іншими студентами .

Четвертий крок полягає у створенні розкладу курсу та навчального плану, у якому викладено очікування для студентів, включаючи терміни виконання завдань та іспитів. Програма також повинна містити інформацію про політику оцінювання, необхідні матеріали та будь-які додаткові ресурси, які можуть бути корисними для студентів.

Після того, як курс розроблено, важливо протестувати його перед тим, як запустити його для студентів. Це може включати залучення бета-тестерів для надання відгуків щодо змісту курсу, тестування функціональності LMS і проведення пілотних класів для усунення будь-яких недоліків.

Нарешті, коли курс готовий до запуску, важливо його ефективно продавати, щоб залучити студентів. Це може включати рекламу через соціальні медіа, кампанії електронною поштою чи інші канали, а також пропонувати заохочення, наприклад знижки або безкоштовні пробні версії .

Доступність: важливо забезпечити, щоб курс був доступним для всіх студентів, у тому числі для людей з обмеженими можливостями. Це може включати надання субтитрів для відео, використання доступних форматів документів для читання та розробку веб-сайту курсу з урахуванням доступності.

Інтерактивність: дистанційні курси повинні бути розроблені таким чином, щоб сприяти взаємодії між студентами та з викладачем. Це може включати дискусійні форуми , живі відеосесії та групові проекти.

Оцінювання: важливо розробляти оцінювання, які точно оцінюють знання студентів і забезпечують своєчасний зворотний зв'язок. Це може включати тести, іспити та оцінені завдання .

Технологія: технологія відіграє вирішальну роль у дистанційній освіті, тому важливо обирати інструменти, які є зручними, надійними та ефективними. Це може включати програмне забезпечення для відеоконференцій , системи керування навчанням і цифрові інструменти для створення та обміну вмістом курсу .

Підтримка інструктора : інструктори повинні бути доступні для надання підтримки та керівництва учням протягом курсу. Це може включати години роботи , підтримку електронною поштою та онлайн-форуми .

Постійне вдосконалення: дистанційні курси слід переглядати та оновлювати на регулярній основі, щоб гарантувати, що вони залишаються актуальними та ефективними. Це може включати отримання відгуків від студентів, аналіз даних курсу та впровадження нових методів і технологій навчання .

Управління часом: Дистанційні курси слід розробляти з урахуванням управління часом, оскільки студенти можуть мати різні пріоритети та обмежений час, який можна присвятити курсу. Це може включати розбивку змісту курсу на керовані частини , надання чітких термінів і пропонування гнучких варіантів планування.

Співпраця: Співпраця є важливим аспектом багатьох курсів, і дистанційні курси повинні бути розроблені таким чином, щоб сприяти співпраці між

студентами. Це може включати групові проекти, експертні оцінки та форуми для спільного обговорення.

Зворотній зв'язок: надання своєчасного та змістовного зворотного зв'язку має вирішальне значення для успіху студента на дистанційному курсі. Це може включати надання зворотного зв'язку щодо завдань і оцінювання, а також можливість студентам отримати відгук від своїх однолітків.

Створення спільноти: Створення почуття спільноти серед студентів є важливим на дистанційному курсі, оскільки це може сприяти залученню та мотивації. Це може включати створення можливостей для студентів спілкуватися один з одним, наприклад, через онлайн-дискусійні форуми або групи в соціальних мережах.

Автентичне навчання : автентичний досвід навчання, такий як реальні проекти чи тематичні дослідження, може допомогти зробити зміст курсу більш актуальним і привабливим для студентів. Це може передбачати включення реальних прикладів і сценаріїв у навчальні матеріали та оцінювання.

Оцінювання для навчання: Оцінювання має не тільки вимірювати знання студентів, але й надавати студентам можливість покращити своє розуміння матеріалу курсу. Це може включати проведення формального оцінювання , наприклад, тестів або практичних завдань , які допомагають учням визначити сфери, де їм потрібна додаткова підтримка.

Персоналізація: персоналізація процесу навчання може допомогти підвищити залученість і мотивацію студентів на дистанційному курсі. Це може включати надання індивідуального зворотного зв'язку , надання студентам можливості вибирати власні завдання чи теми та адаптацію матеріалів курсу відповідно до індивідуальних потреб студентів.

Мультимодальне навчання: використання різноманітних методів навчання може допомогти адаптувати різні стилі навчання та вподобання в дистанційному курсі. Це може передбачати використання поєднання відео, читання, інтерактивних занять і живих сеансів.

Культурна чутливість: Дистанційні курси слід розробляти з урахуванням культурної чутливості, оскільки студенти можуть походити з різного походження та мати різні культурні норми та очікування. Це може включати надання перекладів для матеріалів курсу, надання студентам можливостей поділитися власним досвідом і поглядами, а також пам'ятати про культурні відмінності в стилях спілкування.

Аналіз даних: Аналіз даних курсу може допомогти викладачам визначити сфери, де студенти відчувають труднощі, і відповідно адаптувати матеріали курсу та методи навчання. Це може включати відстеження прогресу студентів, аналіз результатів оцінювання та отримання відгуків від студентів.

Професійний розвиток: Постійний професійний розвиток може допомогти викладачам бути в курсі останніх тенденцій і найкращих практик дистанційної освіти. Це може включати участь у конференціях чи семінарах, співпрацю з колегами та участь у саморефлексії та оцінці.

Служби підтримки: дистанційні курси повинні бути розроблені таким чином, щоб надати студентам доступ до допоміжних послуг, таких як академічні консультації, технічна підтримка та послуги психічного здоров'я. Це може включати партнерство з іншими відділами установи або надання направлень до зовнішніх ресурсів.

Безпека оцінювання: Забезпечення безпеки оцінювання є важливим на дистанційному курсі, оскільки студенти можуть мати більше можливостей шахраювати або брати участь в академічній нечесності . Це може включати використання програмного забезпечення для виявлення плагіату, використання контрольованих іспитів або складання оцінок, які важко обдурити.

Забезпечення якості: Забезпечення якості є важливим аспектом дистанційної освіти, оскільки воно допомагає гарантувати, що курси відповідають потребам студентів і забезпечують високоякісний досвід навчання. Це може включати проведення регулярних оцінок матеріалів курсу та методів викладення , отримання відгуків від студентів і внесення змін на основі цих відгуків.

Навчання викладачів. Забезпечення навчання та підтримки викладачів, які викладають дистанційні курси, є важливими для того, щоб вони були готові надавати високоякісне навчання в цьому форматі. Це може включати навчання з використання технологій, стратегії залучення студентів до дистанційного курсу та передовий досвід дистанційної освіти.

Оцінка курсу: Оцінка ефективності курсу важлива для визначення того, чи відповідає він потребам студентів і чи досягає цілей навчання. Це може включати проведення опитувань або фокус-груп зі студентами, аналіз даних курсу та отримання відгуків від викладачів та інших зацікавлених сторін.

Залучення студентів: підтримка залученості студентів є критично важливим аспектом успішного дистанційного курсу. Це може включати використання інтерактивних елементів , таких як вікторини чи опитування, щоб залучити студентів під час живих сесій, надавши можливості для обговорення та співпраці, а також пропонуючи індивідуальний відгук студентам.

Успіх студента: Забезпечення успіху студента є ключовою метою будь-якого дистанційного курсу. Це може передбачати надання студентам ресурсів і підтримки, щоб допомогти їм подолати труднощі, такі як управління часом або технічні труднощі , а також надання можливостей для виправлення або додаткової підтримки.

Узгодженість програми: Переконайтеся, що дистанційний курс узгоджується з ширшою програмою або навчальним планом, щоб переконатися, що студенти досягають необхідних навчальних цілей і набувають необхідних навичок. Це може включати зіставлення змісту курсу з результатами навчання на рівні програми та перегляд матеріалів курсу та оцінок для узгодження.

Авторське право та інтелектуальна власність : на дистанційному курсі важливо переконатися, що матеріали та зміст курсу відповідають законам про авторське право та інтелектуальну власність. Це може включати отримання дозволу на використання матеріалів, захищених авторським правом, створення оригінального вмісту та надання посилання на джерела.

Доступність: на дистанційному курсі важливо забезпечити доступність матеріалів і змісту курсу для всіх студентів, включно з особами з обмеженими можливостями. Це може включати надання альтернативних форматів для матеріалів курсу, таких як аудіоописи чи стенограми, а також розробку веб-сайту курсу з урахуванням доступності. Для успішної розробки дистанційного курсу з об'єктно-орієнтованого програмування потрібно планування якісних навчальних матеріалів, зручної системи керування навчанням, ретельного тестування та ефективного маркетингу. Крім того, важливо звернути увагу на доступність, інтерактивність, оцінювання, технології, підтримку викладача та постійне вдосконалення. Використовуючи найкращі практики дистанційної освіти, викладачі можуть створювати привабливі та ефективні умови навчання для студентів, незалежно від їхнього місцезнаходження.

Щоб показати структуру навчального курсу ми створили таблицю на три стовпчики: "№", "Навчальні цілі", "Очікувані результати". Кількість рядочків в таблиці дорівнює кількості модулів(блоків) курсу + 1 рядок для навчальних цілей та очікуваних результатів курсу.

Заповнена табл. 2.1 з навчальними цілями та очікуваними результатами курсу, а також з результатами кожного модуля(блоку) курсу.

Таблиця 2.1 – Структура курсу: Основи об'єктно-орієнтованого програмування

№	Навчальні цілі	Очікувані результати
Розділ 1. Основи синтаксису мови Java.	Вивчити основи синтаксису мови Java, основні конструкції мови, підключення та використання бібліотек.	Студенти повинні вміти розробляти програми з використання простих синтаксичних конструкцій. Вміти підключати та використовувати вбудовані бібліотеки мови Java.
Розділ 2. Керуючі конструкції мови Java.	Вивчити стандартні алгоритмічні структури (лінійні, розгалужені. циклічні);	Студенти повинні вміти розробляти програми з використанням умовних та циклічних конструкцій. Здійснювати вибір алгоратмів виконання програми.
Розділ 3. Робота зі структурованими даними.	Вивчити принципи роботи зі структурованими даними (робота з масивами та списками)	Студенти повинні вміти використовуючи циклічнаі та умовні конструкції працювати з масивами та списками
Розділ 4. Розробка	Вичити основи роботи з файлами	Студенти повинні вміти створювати

програм з використанням файлів та баз даних.	та базами даних, основами роботи з графічним інтерфейсом.	програми з можливостями обміну даними з файлами та базами даних. Повинні розробляти програми з графічним інтерфейсом.
За курс:	Вивчити основи програмування в середовищі Java.	Студенти повинні вміти розробляти прості програми, здійснюючи правильний вибір типів даних. Вміти підключати та використовувати бібліотеки, файли та бази даних. Вміти відлагоджувати програмне забезпечення, здійснювати пошук у мережі Internet.

На рис. 2.1 показаний результат розміщення навчальних цілей та очікуваних результатів курсу та кожного модуля у відповідні блоки курсу.

Рисунок 2.1 – Розміщення навчальних цілей та очікуваних результатів курсу.

2.2 Розробка засобів контролю засвоєння матеріалу

У сучасному світі, коли навчання та освіта залишаються все більшими вимогами для успішного життя, контроль засвоєння матеріалів стає достатньою. Цей контроль може бути дієвим такими ефективними способами, як

традиційні тести та екзаменів до більш сучасних методів, як оцінка проектів та електронні тести.

Одним із найпоширеніших методів контролю засвоєння матеріалу є традиційні тести та екзамени. Ці методи традиційно полягають у запитаннях, які мають правильну відповідь, або завданнях, де студентам потрібно виконати деякі та відповіді на питання. Однак ці методи не завжди ефективні, оскільки вони можуть бути підшивані та шахрайству, часто не показують повного розуміння матеріалу та не враховують індивідуальні рівні знань та навичок.

Одним із більш сучасних методів контролю навчального матеріалу є електронні тести та оцінка проектів. Електронні тести можуть бути більш точними та об'єктивними, оскільки вони можуть вимірювати більше аспектів знань та навичок студентів. Оцінка проектів може бути більш зручною та цікавою для студентів, оскільки вони можуть використовувати свої унікальні навички та таланти для того, щоб довести свої знання.

На додаток до методів, згаданих у попередньому розділі, існує кілька інших підходів, які можна використовувати для моніторингу прогресу навчання. Наприклад, формувальне оцінювання, яке є безперервним процесом збору та використання доказів для покращення навчання, можна використовувати для моніторингу прогресу студента протягом курсу чи програми. Цей підхід передбачає надання студентам зворотного зв’язку щодо їх успішності, визначення областей, де їм може знадобитися додаткова підтримка чи ресурси, і коригування навчання для кращого задоволення їхніх потреб.

Іншим підходом є самооцінка, коли студентів заохочують обмірковувати власне навчання та прогрес. Це можна зробити за допомогою різних інструментів, таких як навчальні щоденники , роздуми або тести для самооцінки. Самооцінка може бути потужним інструментом для сприяння метапізнанню та допомоги учням у кращому розумінні власних процесів навчання.

В останні роки зростає інтерес до використання технологій для розробки інструментів для моніторингу прогресу навчання. Наприклад, аналітику навчання, яка передбачає аналіз даних про успішність і поведінку учнів, можна

використовувати для виявлення моделей і тенденцій у навчанні учнів. Потім ця інформація може бути використана для прийняття рішень щодо навчання та покращення результатів навчання .

Технології адаптивного навчання є ще одним прикладом технологічних інструментів для моніторингу прогресу навчання. Ці інструменти використовують дані про успішність учнів, щоб налаштувати навчання та забезпечити персоналізований навчальний досвід, адаптований до індивідуальних потреб і здібностей кожного учня.

Однією з важливих міркувань при розробці інструментів для моніторингу прогресу навчання є необхідність збалансувати формувальне та підсумкове оцінювання. Формуюче оцінювання зосереджено на зборі постійного зворотного зв'язку для покращення навчання, тоді як підсумкове оцінювання зосереджено на оцінці успішності студента наприкінці курсу чи програми. Обидва типи оцінювання є важливими, але важливо забезпечити баланс між ними, щоб учні мали постійні можливості вдосконалювати своє навчання, а також несли відповідальність за свою успішність.

Іншим міркуванням є необхідність використання різних методів оцінювання, щоб гарантувати, що студентів оцінюють за низкою навичок і навичок. Наприклад, у той час як тести та вікторини можуть бути ефективними для оцінювання знань і запам'ятовування, інші методи, такі як оцінювання продуктивності, портфоліо та проекти, можуть бути краще підходящими для оцінювання навичок мислення вищого рівня, креативності та здібностей до вирішення проблем.

Крім того, важливо переконатися, що методи оцінювання узгоджуються з цілями та результатами навчання. Це означає, що завдання оцінювання повинні бути розроблені для оцінки конкретних навичок і знань, які студенти повинні засвоїти, і що критерії оцінювання чітко повідомляються студентам, щоб вони розуміли, що від них очікується.

Що стосується технологічних інструментів для моніторингу прогресу навчання, важливо переконатися, що ці інструменти розроблені з урахуванням

потреб і вподобань студентів. Наприклад, технології адаптивного навчання повинні бути розроблені для забезпечення персоналізованого досвіду навчання, який буде захоплювати та мотивувати студентів, тоді як інструменти аналітики навчання мають бути розроблені, щоб надавати практичну інформацію, яку можна використовувати для покращення результатів навчання.

Нарешті, важливо залучати студентів до процесу оцінювання, надаючи їм можливість обміркувати власне навчання та прогрес. Це може допомогти учням розвинути краще розуміння власних процесів навчання та може сприяти почуттю власності та відповідальності за власне навчання. Залучаючи студентів до процесу оцінювання та надаючи постійний зворотній зв'язок і підтримку, викладачі можуть допомогти студентам повністю реалізувати свій потенціал і переконатися, що вони просуваються до своїх навчальних цілей.

Однією з важливих міркувань при розробці інструментів для моніторингу прогресу навчання є використання рубрик. Рубрики — це набір критеріїв, які використовуються для оцінювання виконання учнями певного завдання чи завдання. Вони особливо корисні для оцінювання складніших навичок, таких як критичне мислення та вирішення проблем, які важко виміряти за допомогою традиційних тестів або вікторин. Рубрики можна використовувати для визначення чітких очікувань щодо успішності студентів, допомогти студентам зрозуміти, як їх оцінюватимуть, і надати зворотній зв'язок, який є конкретним і дієвим.

Іншим важливим моментом є використання зворотного зв'язку. Зворотний зв'язок є критично важливим компонентом формувального оцінювання, оскільки він надає учням інформацію про їх успішність і допомагає їм визначити сфери, де вони можуть покращитися. Ефективний зворотний зв'язок має бути своєчасним, конкретним і дієвим, і має бути спрямований на підтримку студентів у навчанні, а не просто на оцінку їхньої роботи.

Крім того, важливо враховувати роль залучення студентів у моніторинг прогресу навчання. Студенти, які залучені до навчання, швидше за все, будуть мотивовані та інвестовані у власний прогрес, що може призвести до кращих результатів навчання. Один із способів сприяти залученню – це надати

учням можливість взяти на себе відповідальність за своє навчання, наприклад, за допомогою самооцінки або проектного навчання.

У процесі розробки ефективних засобів контролю захисту матеріалу важливо встановити цілі та об'єктиви контролю, щоб використовувати найбільш ефективні методи. Загалом, використання різноманітних підходів до моніторингу прогресу навчання, зокрема традиційних тестів, формувального оцінювання, самооцінювання та технологічних інструментів, допомагає викладачам отримати більш повне розуміння того, як учні навчаються, і скорегувати свої інструкції для кращого відповідання їхнім потребам. Однак, важливо пам'ятати, що моніторинг прогресу навчання є безперервним процесом, який вимагає постійного оцінювання та вдосконалення. Регулярне оцінювання ефективності методів оцінювання та відкритість до зворотного зв’язку від студентів допомагає викладачам покращувати навчальний процес та забезпечити найкращий навчальний досвід для своїх учнів.

3 РОЗРОБКА ANDROID ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНОГО НАВЧАННЯ З ДИСЦИПЛІНИ "ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ"

3.1 Розробка мобільного додатку для дистанційного навчання

Зменшення фінансових витрат через використання найсучасніших інформаційних та мобільних технологій та поширення дистанційного навчання у освіті, використання віртуальних класів, поширення впливу практиків на навчання через мережу. Всі ці світові тенденції вимагають від викладачів постійно підвищувати рівень своєї кваліфікації як професійної так і технологічної та перегляда методики проведення навчального процесу.

Вчителям посильно створення дистанційної підтримки навчання, що характеризується доступністю, простотою, можливістю використовувати наявні ресурси та створювати власні розробки з використанням популярних соціальних сервісів.

Дистанційне навчання відбувається з використанням LMS (Learning Management System - Система Управління Навчанням), у якій створюються інтерактивні курси або уроки. Навчання проходить під керівництвом тьютора, а вчитель виступає як наставник та координатор (локальний тьютор).

З цією метою у педагога повинні бути сформовані такі вміння:

1. Інтернет пошук і дослідження - педагоги повинні знати, як зробити правильний інтернет-пошук, використовуючи пошукові терміни та модифікатори.
2. Використання додатків для підвищення продуктивності. Педагоги повинні знати, як створювати, редагувати й змінювати документи.
3. Отримання технічної допомоги. Пошук безкоштовних ресурсів.
4. Соціальні сервіси. Як правильно використовувати соціальні сервіси для навчання і роботи, як захистити себе, питання кіберзалякування, спілкування з іншими людьми вашої професії (персональна навчальна мережа).
5. Цифрове громадянство. Етикет соціальних сервісів. Як правильно використовувати Інтернет, написати повідомлення або блог.

6. Охорона і безпека. Педагоги повинні знати про антивіруси, спам, фішинг для свого власного захисту і навчити цьому своїх студентів.

7. Основи роботи з обладнанням та усунення неполадок. Знати назви різних технологій, як виконати невеликий ремонт, можливі несправності Wi-Fi, мережі, операційної системи і т. ін.

8. Резервне копіювання даних.

9. Пошук програм та програмного забезпечення. Як знайти, оцінити і використати додатки для навчального процесу.

10. Авторське право та посилання на джерела.

11.Перед пошуком необхідно визначити, що необхідно знайти. Необхідно шукати предмет пошуку, а не саме питання

12. Створіть Google Alerts. Якщо пошук за деякими темами здійснюється часто, це сервіс буде оповіщати про появу нової інформації.

13. Google Books є гарним місцем для пошуку книг і журналів, які можна переглянути безкоштовно в Інтернет і зберегти на книжковій полиці, доступ до якої можна дати студентам.

3.2 Розробка бази даних матеріалів дистанційного курсу

У сучасному світі, що базується на даних, бази даних стали основою різних додатків і систем. Від управління великими обсягами інформації до забезпечення цілісності та доступності даних, бази даних відіграють важливу роль у зберіганні, організації та пошуку даних.

За своєю суттю, база даних - це організована колекція структурованих даних, які зберігаються і управляються в електронному вигляді. Вона є центральним сховищем зберігання інформації, до якої можна легко отримати доступ, керувати нею та оновлювати. Основна мета бази даних - забезпечити ефективний і надійний спосіб зберігання та пошуку даних, гарантуючи їх цілісність, безпеку та масштабованість.

Бази даних можна класифікувати на різні типи на основі моделей і структур даних, що лежать в їх основі. Деякі з найпоширеніших типів включають:

Реляційні бази даних: Реляційні бази даних організовують дані в таблиці із заздалегідь визначеними зв'язками. Вони використовують мову структурованих запитів (SQL) для управління та маніпулювання даними. Популярні системи управління реляційними базами даних (СКБД) включають MySQL, Oracle і Microsoft SQL Server.

Бази даних NoSQL: Бази даних NoSQL відрізняються від традиційної реляційної моделі і надають гнучкі схеми для зберігання неструктурованих або напівструктурованих даних. Вони відмінно справляються з великими обсягами даних і підтримують розподілені та масштабовані архітектури. Прикладами баз даних NoSQL є MongoDB, Cassandra та Redis.

Об'єктно-орієнтовані бази даних: Об'єктно-орієнтовані бази даних зберігають дані в об'єктно-орієнтованих форматах, що дозволяє безпосередньо зберігати та отримувати складні структури даних. Вони підходять для додатків, які значною мірою покладаються на об'єктно-орієнтовані парадигми програмування.

У сфері сучасної розробки програмного забезпечення зв'язок між базами даних та інтерфейсними додатками є вирішальним кроком у створенні надійних та динамічних систем.

Перш ніж описати процес підключення, необхідно зрозуміти природу взаємодії інтерфейсу і бази даних. Інтерфейс, що складається з HTML, CSS та JavaScript, відповідає за представлення користувацького інтерфейсу та обробку взаємодії з користувачем. З іншого боку, база даних зберігає та управляє даними програми.

Найпоширенішим підходом для підключення баз даних до інтерфейсних додатків є використання внутрішнього сервера як посередника. Бекенд діє як інтерфейс між інтерфейсом і базою даних, забезпечуючи пошук даних, маніпуляції з ними та іншу бізнес-логіку. Такий підхід має низку переваг, зокрема підвищену безпеку, контрольований доступ до даних і розподіл проблем між інтерфейсом і базою даних.

Найпоширенішим методом зв'язку інтерфейсу з бекендом і базою даних є використання API (інтерфейсів прикладного програмування). API визначають

набір правил і протоколів для зв'язку між різними програмними компонентами. Розробляючи та впроваджуючи RESTful API, бекенд може відкривати кінцеві точки, які дозволяють фронтенду запитувати та отримувати дані з бази даних у структурований та контрольований спосіб.

Для підключення внутрішнього сервера до бази даних можна використовувати різні методи і технології, залежно від конкретної системи управління базами даних (СУБД), що використовується. Деякі загальні підходи включають:

Драйвери для конкретної бази даних: Більшість СУБД надають власні драйвери, які полегшують зв'язок між внутрішнім сервером і базою даних. Ці драйвери, такі як JDBC для додатків на основі Java або ODBC для з'єднань загального призначення, дозволяють розробникам взаємодіяти з базою даних за допомогою стандартизованих API.

Інструменти об'єктно-реляційного відображення (ORM): Інструменти ORM абстрагують взаємодію з базою даних, дозволяючи розробникам працювати з об'єктно-орієнтованими представленнями даних вищого рівня.

При обміні даними між бекендом і фронтендом дуже важливо вибрати відповідні формати обміну даними. Найпоширеніші формати включають JSON (JavaScriptObjectNotation): JSON - це легкий, зрозумілий людині формат, який широко підтримується сучасними веб-технологіями. Він добре підходить для передачі структурованих даних між бекендом і фронтендом, дозволяючи легко аналізувати та маніпулювати ними в JavaScript.

XML (eXtensibleMarkupLanguage): XML - це гнучка мова розмітки, яка використовується для представлення та обміну даними. Хоча вона менш поширена в сучасній веб-розробці, вона залишається життєздатним варіантом, особливо в сценаріях, що вимагають більш складних структур даних або застарілих систем.

Для з'єднання інтерфейсу з бекендом і базою даних розробники можуть використовувати фреймворки і бібліотеки, які спрощують пошук даних і маніпуляції з ними.

Схожість

Джерела з Бібліотеки 85

1	Студентська робота	ID файлу: 1015192133	Навчальний заклад: Lviv Polytechnic National University	38 Джерело	0.38%
2	Студентська робота	ID файлу: 1015226590	Навчальний заклад: National Technical University of Ukr	13 Джерело	0.25%
3	Студентська робота	ID файлу: 1015196056	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.22%
4	Студентська робота	ID файлу: 1014817889	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.21%
5	Студентська робота	ID файлу: 1015230312	Навчальний заклад: Lviv Polytechnic National University	3 Джерело	0.14%
6	Студентська робота	ID файлу: 1007493609	Навчальний заклад: Taras Shevchenko National University of Kyiv		0.13%
7	Студентська робота	ID файлу: 1014894954	Навчальний заклад: Taras Shevchenko National Universit	3 Джерело	0.13%
8	Студентська робота	ID файлу: 1015219049	Навчальний заклад: State University Kyiv National Economic Univ...		0.13%
9	Студентська робота	ID файлу: 1015109373	Навчальний заклад: Vasyl Stus Donetsk National University		0.12%
10	Студентська робота	ID файлу: 1014912284	Навчальний заклад: State University Kyiv National Econo	4 Джерело	0.12%
11	Студентська робота	ID файлу: 1015224745	Навчальний заклад: Lviv Polytechnic National University	4 Джерело	0.1%
12	Студентська робота	ID файлу: 1009636404	Навчальний заклад: Lviv Polytechnic National University	3 Джерело	0.1%
13	Студентська робота	ID файлу: 1015198601	Навчальний заклад: National Aviation University		0.1%
14	Студентська робота	ID файлу: 1015118023	Навчальний заклад: National Aviation University	3 Джерело	0.1%
15	Студентська робота	ID файлу: 1015080220	Навчальний заклад: National Technical University of Ukra	2 Джерело	0.1%
16	Студентська робота	ID файлу: 1015196499	Навчальний заклад: National Technical University of Ukraine "Kyi...		0.1%
17	Студентська робота	ID файлу: 1008189121	Навчальний заклад: Taras Shevchenko National Universit	3 Джерело	0.1%
18	Студентська робота	ID файлу: 1014274157	Навчальний заклад: National University Ostroh Academy		0.1%
19	Студентська робота	ID файлу: 1011073987	Навчальний заклад: Lviv Polytechnic National University		0.1%