ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка моделі додатку для моделювання випадкових музичних послідовностей з реалізацією на базі Swift»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Сергій ТОКАРЄВ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Сергій ТОКАРЄВ
група ІСДМ-63

Керівник: Андрій БОНДАРЧУК
науковий ступінь, вчене звання д.т.н., професор

Рецензент:
науковий ступінь, вчене звання
_____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Токарєву Сергію Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка моделі додатку для моделювання випадкових музичних послідовностей з реалізацією на базі Swift

керівник кваліфікаційної роботи Андрій БОНДАРЧУК д.т.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, параметри систем моніторингу, вимоги до IoT мереж.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз принципів функціонування програмного забезпечення для створення музикальних послідовностей

Дослідження сучасних інструментів розробки програмного забезпечення для створення музикальних послідовностей

Розробка програмного забезпечення для створення випадкових музичних послідовностей з реалізацією на базі Swift

5. Перелік графічного матеріалу: *презентація*
    1. Вправи для розвитку музичного слуху
    2. Вибір платформи для розробки додатку
    3. Формат MIDI
    4. Алгоритм генерації наступного інтервалу для розпізнавання
    5. Основні екрани додатку

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-05.11.23	
2	Аналіз принципів функціонування програмного забезпечення для створення музикальних послідовностей	05.11-12.11.23	
3	Дослідження сучасних інструментів розробки програмного забезпечення для створення музикальних послідовностей	13.11-19.11.23	
4	Розробка програмного забезпечення для створення випадкових музичних послідовностей з реалізацією на базі Swift	20.11-25.11.23	
5	Розробка архітектури додатку	27.11-03.12.23	
6	Економічне обгрунтування	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка демонстраційних матеріалів	21.12-29.12.23	

Здобувач вищої освіти _____ Сергій ТОКАРЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій БОНДАРЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 53 стор., 4 табл., 18 рис., 18 джерел.

Мета роботи – розробка додатку для моделювання випадкових музичних послідовностей мовою Swift

Об'єкт дослідження – процес моделювання моделювання випадкових музичних послідовностей

Предмет дослідження – методи та засоби моделювання, платформа для розробки ПЗ мовою Swift

Короткий зміст роботи: У роботі проведено програмних засобів навчання музиці. Проаналізовано основні принципи розробки додатків та здійснено порівняння програмного забезпечення для створення музикальних послідовностей. Розроблено програмнео забезпечення для створення випадкових музичних послідовностей з реалізацією на базі Swift.

КЛЮЧОВІ СЛОВА: ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, АРХІТЕКТУРА, ДОДАТОК, SWIFT, МОДЕЛЮВАННЯ.

ABSTRACT

Text part of the master's qualification work: 53 pages, 18 pictures, 4 table, 18 sources.

The purpose of the work : development of an add-on for modeling random musical sequences in my Swift

Object of research – process of modeling of random musical sequences

Subject of research – methods for designing a model, a platform for developing software development software Swift

Summary of the work: The robot has carried out a program for learning music. The basic principles of development of accessories have been analyzed and the leveling of software for creating musical sequences has been developed. Software has been developed for creating unique musical sequences based on Swift.

KEYWORDS: SOFTWARE, ARCHITECTURE, ADDENDUM, SWIFT, MODELING.

ЗМІСТ

ВСТУП

Навчання музиці є дуже популярним. Станом на зараз на ринку навчання є дуже багато варіантів, як державних установ: шкіл, коледжів, ЗВО, так і приватних студій та індивідуальних викладачів.

Непрофесійне навчання музиці пов’язане з деякими труднощами. Насамперед, аматори мають небагато часу для занять, тому їх уроки переважно складаються з безпосереднього оволодіння інструментом або голосом, і майже ніколи не включають у себе додаткових навичок.

Тим часом, такі додаткові навички є дуже важливими для успішного навчання, і однією з найважливіших є розвиток музичного слуху.

Здобувачі професійної музичної освіти на протязі усього навчання проходять курс сольфеджіо. В той же час непрофесіональні музиканти майже ніколи цього не вивчають, що вкрай негативно впливає на їх професійні успіхи.

Причиною відмови від вивчення сольфеджіо для аматорів є наступне:

- Брак часу. Курс сольфеджіо у базовому варіанті потребує приблизно 150 годин, що є три уроки на тиждень впродовж року.
- Складність матеріалу. Навіть базові поняття з теорії музики є досить складними для вивчення
- Потребує практичних занять. Музичний слух неможливо розвинути, вивчаючи теорію, необхідно виконувати складні вправи: розпізнавання на слух, диктанти, спів мелодій тощо.

Для тих учнів, хто вивчає сольфеджіо, критичною є регулярність занять та самостійної роботи. У той час як деякі вправи потребують спеціальної обстановки: тиші, наявності інструменту тощо, деякі вправи можна виконувати у будь який вільний час.

Однією з таких вправ є розпізнавання музичних інтервалів на слух. Це базова вправа, яка потребує мінімальної теорії, і яка не потребує спеціальної обстановки. У той же час ця вправа є дуже корисною для розвитку

гармонійного слуху. Регулярне виконання дозволить покращити інтонування при співі, краще чути помилки при грі на інструменті, а також краще розпізнавати та підбирати на слух музичні мелодії.

Для виконання цієї вправи ідеально підходить мобільний пристрій. У даній роботі буде розглянуто підхід до створення додатку для розпізнавання музичних інтервалів на слух та реалізовано додаток для iPhone.

1 АНАЛІЗ ПРИНЦИПІВ ФУНКЦІОНУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ МУЗИКАЛЬНИХ ПОСЛІДОВНОСТЕЙ

Розуміння теорії музики означає, що у вас є інструменти, щоб складати свою музику, а також краще розуміти та цінувати музику загалом. За допомогою програм та сервісів можна освоїти теорію музики та розпочати своє власне музичне навчання.

Хоча вам не потрібно нічого знати про теорію музики, щоб бути хорошим музикантом або отримувати задоволення від музики, вивчення теорії дає безліч переваг, і вам не потрібно записуватися в музичну школу, щоб вивчати її. Він може перетворити вас на більш досконалого музиканта, допомогти вам навчитися читати ноти, спростити твір власної музики, а також допомогти вам зрозуміти та насолоджуватися музикою, яку ви слухаєте щодня.

По суті, кожен музичний твір складається із двох основних речей: висоти звуку та ритму. Вивчення того, як ці концепції працюють і розвиваються власними силами, необхідне для засвоєння теорії. Найкращі курси для вивчення теорії музики охоплюють фундаментальні та просунуті концепції та використовують просту мову та корисні діаграми чи аудіо-приклади для навчання. Ось основні речі, які ви повинні очікувати від будь-якого навчального ресурсу з теорії музики:

- **Пітч-тренінг:** Кращі програми та веб-сайти для навчання теорії музики повинні пояснювати, що таке висота звуку, і спиратися на нього за допомогою таких тем, як гами, інтервали, акорди, інверсії, тональності, гармонія, режими та природним чином переходити до більш складним елементам висоти звуку, таким як септаккорди та неаполітанські акорди.
- **Ритмічне тренування:** Ритм допомагає формувати ноти та визначає тривалість кожної. Кожна програма повинна розповідати вам про всі тривалості, ритмічні структури та темпи, а також про тимчасові розміри, які підтримують порядок. Деякі програми можуть навіть навчити вас ритмічним

символам та цінностям та навчити вас самостійно відбирати ритми та поліритми, щоб розвивати свої навички роботи з ними. Відмінні курси можуть також охоплювати ідеї та техніки композиції, що використовуються імпресіоністами та сучасними композиторами, оркестрування та інші другорядні, але все ж таки актуальні концепції.

- **Тренування вуха:** Саме так це звучить, тренуючи свій музичний слух, і це є важливою частиною вивчення теорії музики. У цьому процесі ви навчитеся визначати на слух такі речі, як інтервали, типи акордів, модуляції та каденції, просто чуючи їхню гру і не дивлячись на свої ноти. Так само деякі програми можуть також навчати співу з аркуша, що змушує вас співати кілька нот (а не всю пісню), що допомагає вам покращити вашу висоту звуку та точність ритму, що покращує ваше розуміння теорії музики. Звичайно, це може здатися дивним або непотрібним, але це дійсно допомагає вам швидко виявляти речі, виявляти помилки і висувати сильніші ідеї (як у Amadeus кліп, посилання на який вище).
- **Вправи та вікторини:** Хороші програми з теорії музики також повинні надавати безліч можливостей попрактикуватися та перевірити те, що ви вивчаєте на кожному уроці, щоб закріпити концепції у вашому розумі, перш ніж переходити до наступного. Вам буде корисно мати фізичну або віртуальну клавіатуру, а також рукописний папір для запису нотаток під час цих уроків і навіть для самостійних експериментів.

1.1 Функціональні характеристики EarMaster

EarMaster — це музичний додаток для Windows, Mac, iOS і Android, запущений у 1996 році датським редактором Miditec, який змінив назву на EarMaster ApS у 2005 році. Перший прототип програмного забезпечення був заснований на DOS, але з 1996 року він був адаптований під кілька операційних систем. Основна увага EarMaster — тренування слуху та сприйняття слуху, хоча EarMaster, має тенденцію до більш загального підходу до викладання музики з версії 4.0, охоплюючи ширший спектр музичної теорії та практики.

EarMaster включає в себе кілька режимів навчання: семінари з прогресивними наборами уроків з різних тем з теорії музики та навчання слуху, режим налаштування та курс для початківців. У той час як загальні майстер-класи є досить загальними, і їхні уроки зосереджені на більшості аспектів тренування слуху, джазові семінари зосереджуються виключно на особливостях джазової музики (наприклад, джазовий акорд, ритми свінгу та справжній спів з книжок). За допомогою режиму налаштування користувач може налаштувати власні вправи для індивідуальної практики. Теми, які охоплює EarMaster, це інтервальний спів, порівняння інтервалів, ідентифікація інтервалів, ідентифікація гами, ідентифікація акордів, ідентифікація інверсії акордів, ідентифікація прогресії акордів, диктування ритму, читання ритму (читання з виду), плескання ритму, виявлення ритмічних помилок, мелодичний диктант , мелодійний спів і мелодійний спів.

Відповіді на запитання надаються за допомогою екранних інтерфейсів (софт, фортепіано, гітара, бас, скрипка, віолончель, банджо та інші струнні інструменти), функціональної клавіатури зі шкалою ступенів і складів для сольфеджі, кнопок із вибором кількох варіантів, MIDI- інструменту або через мікрофон (голос, плескаючі або акустичні інструменти).

Користувач може вибирати між кількома системами іменування нот для виконання вправ: літери (A, B, C тощо), градуси шкали , сольфедж із фіксованим виконанням або відносне сольфеджування , що робить його сумісним із методом Кодалі .

Результати кожного уроку фіксуються та аналізуються у вікні статистики. Їх можна автоматично синхронізувати через систему EarMaster Cloud для миттєвого доступу вчителів музики. EarMaster Cloud — це хмарна система для музичних шкіл, розроблена EarMaster ApS. Він поєднує онлайн-керування ліцензіями та хмарну синхронізацію завдань і результатів студентів.

EarMaster поширюється в електронному вигляді для завантаження. Програмне забезпечення сумісне з Microsoft Windows , MacOS , iOS і Android (включаючи Chromebook).

Користувач може вводити відповіді за допомогою пристрою MIDI , відтворюючи свої відповіді. Програмне забезпечення також включає визначення висоти в реальному часі , що дозволяє користувачам співати або відтворювати свої відповіді та отримувати негайну оцінку їх висоти та ритмічної точності.

Звуки, які відтворює програмне забезпечення, створюються механізмом відтворення SoundFont і охоплюють повний набір попередніх налаштувань звуку General Midi (GM).

Починаючи з версії 6.1, EarMaster може імпортувати музичні партитури до 8 голосів у форматі Music XML , щоб використовувати їх у своїх ритмічних і мелодичних вправах (наприклад , для співу SATB).

EarMaster містить редаговану бібліотеку з акордами, гамами, прогресією акордів і понад 600 партитурами джазової та класичної музики, які студенти та вчителі можуть використовувати у спеціальних вправах і завданнях.

Додаток має багато функцій, орієнтований на вивчення музичної теорії, навчання організовано у вигляді курсів та вправ до них. Потребує систематичного підходу, має декілька розділів для тренування різних навичок.

Переваги:

- Широкий вибір вправ
- Тренування різних навичок

Недоліки:

- Потребує систематичного підхода
- Умовно-безплатний додаток (покупки всередині додатку)

1.2 Функціональні характеристики Tenuto

Tenuto Додаток для всеосяжного розвитку музичних навичок: абсолютного слуха, розпізнавання інтервалів, акордів та тональностей для

ваших iPhone, iPad та iPod touch. Кожна з 24 вправ використовує зручний сенсорний інтерфейс, який повністю оптимізований для всіх розмірів екрана на всіх пристроях.

Tenuto не потребує доступу до Інтернету — продовжуйте навчання музиці, сидячи в автобусі або відпочиваючи на пляжі. Повернувшись додому або поблизу точки доступу Wi-Fi, надішліть звіт про виконання вправи своєму вчителю.

Функціонал:

• Ідентифікація ноти
• Ідентифікація тональності
• Ідентифікація інтервалу
• Ідентифікація звукоряду
• Ідентифікація акорду
• Побудова нот
• Побудова тональності
• Побудова інтервалів
• Побудова гами
• Побудова акордів
• Зворотна ідентифікація клавіатури
• Ідентифікація ноти клавіатури
• Ідентифікація інтервалу клавіатури
• Ідентифікація звукоряду клавіатури
• Ідентифікація акордів клавіатури
• Ідентифікація ноти на грифі
• Ідентифікація інтервалу на грифі
• Ідентифікація звукоряду на грифі
• Ідентифікація акордів на грифі
• Тренування клавіатури на слух
• Тренування нот на слух
• Тренування інтервального слуху

• Тренування гамми

• Тренування акорду

Рисунок 1.1 Інтерфейс додатоку Tenuto

Остання версія Версія 4.2. В ній додано підтримку темного режиму. У темному режимі; меню, діалогові вікна та параметри мають темний вигляд. Великі білі області, наприклад фон нотного запису або білих клавіш піаніно, затемнені.

Збільшує розташування кнопок відповіді на певних пристроях для кращого узгодження з системною клавіатурою.

Вирішено проблему затримки звуку під час використання певних навушників Bluetooth.

Виправляє різні незначні проблеми макета.

Переваги додатку:

- Широкий вибір вправ
- Багато вправ для розвитку гармонійного слуху
- Вправи на розвиток абсолютного слуху

Недоліки:

- Платний додаток ($4.99)
- Доступні лише звуки фортепіано

1.3 Функціональні характеристики Solfeggio Education

Solfeggio Education (Сольфеджіо) - це професійний додаток для спектрального аналізу аудіосигналів, розвитку музичних навичок, тонкого настроювання інструментів, а також серйозних наукових досліджень та навчання студентів. Програма в режимі реального часу захоплює аудіосигнал з різних джерел (мікрофон, генератор ідеальних тонів), після чого розраховує його точний спектр, при цьому виділяючи музичні ноти, що звучать з високою точністю.

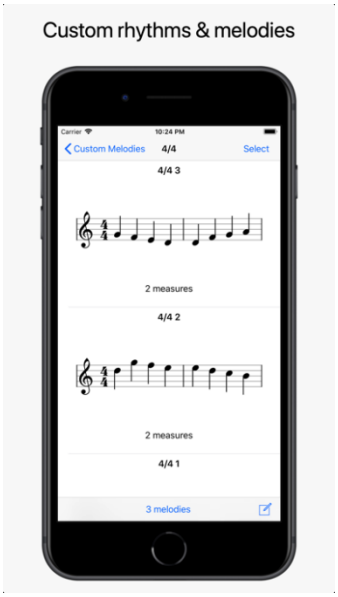
Вирізняється поміж інших тим, що серед вправ є мелодійні диктанти.

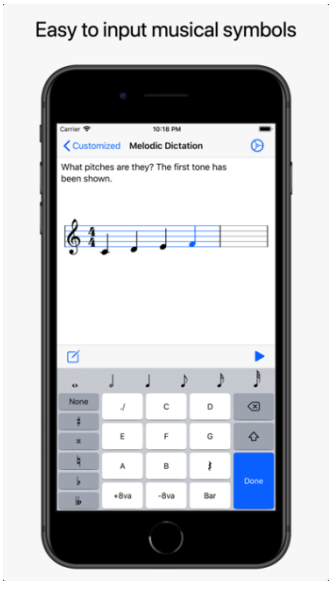
Переваги:

- Багато вправ (понад 100 схем)
- Вправи, які недоступні у аналогах
- Вправи на точність співу (аналізує ноту, яку співають, у реальному часу через мікрофон)
- дизайн, адаптований під iOS 7
- зручний пошук за темами

Недоліки:

- Платний додаток
- Тільки для iPad





-

- Рисунок 1.2 Інтерфейс додатку Solfeggio Education

2 ДОСЛІДЖЕННЯ СУЧАСНИХ ІНСТРУМЕНТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ МУЗИКАЛЬНИХ ПОСЛІДОВНОСТЕЙ

2.1 Swift – нативна для платформи iOS мова програмування

Swift – це потужна та ефективна мова програмування, яка є нативною для платформи iOS, що робить її ключовим інструментом для розробників мобільних додатків. Розробка на Swift дозволяє створювати високоякісні, швидкі та стабільні програми, які оптимально взаємодіють з усіма функціями та можливостями iOS. Однією з переваг Swift є його висока продуктивність, яка досягається завдяки оптимізаціям та високоякісному виконанню коду. Мова програмування Swift має сучасний синтаксис та пропонує широкий спектр інструментів, що спрощує процес розробки та підтримку програм. Swift активно підтримується та оновлюється розробниками, що забезпечує стабільність та сумісність з новими версіями операційної системи iOS. Інтеграція Swift із системними функціями iOS, такими як Core Data, Core Graphics та Core Animation, робить його важливим інструментом для розробки різноманітних додатків. Swift дозволяє ефективно використовувати мультимедійні можливості iOS, забезпечуючи високу якість графіки та звуку в додатках. Інтегроване середовище розробки Xcode сприяє комфортному програмуванню на Swift, забезпечуючи багатофункціональний інтерфейс та зручний дебагінг. Swift підтримує парадигму програмування "безпеки за замовчуванням" (default safe programming), що зменшує ймовірність помилок під час розробки. За допомогою Swift можна легко інтегрувати різноманітні бібліотеки та фреймворки для розширення функціональності додатків. Мова має високу швидкість виконання, що особливо важливо для розробки вимогливих до ресурсів додатків, таких як ігри чи графічні редактори. Swift дозволяє розробникам використовувати різні парадигми програмування, включаючи об'єктно-орієнтоване та функціональне програмування. Розширена підтримка мови Swift у спільноті розробників сприяє обміну знаннями та швидкому

вирішенню проблем. Swift є ідеальним вибором для розробки додатків для всіх пристроїв Apple, починаючи від iPhone та iPad, закінчуючи Mac та Apple Watch. Оглядаючи всі аспекти, можна визначити, що Swift є не тільки мовою програмування, а й потужним інструментом для створення інноваційних та високоякісних додатків на платформі iOS.

Якщо розглядати альтернативи, є декілька доступних:

- Objective C
- Flutter/Kotlin Multiplatform/React Native
- Web View-based підхід

Розглянемо їх детальніше.

Objective C

Був основною мовою розробки під платформу Apple до появи Swift. Має менш зручний і дещо незвичний синтаксис, набагато складніший в опануванні, ніж Swift, також не має першокласної підтримки сучасних технологій програмування, таких як асинхронне програмування. Також з Objective C неможливо використовувати сучасний підхід з описом інтерфейса користувача за допомогою коду.

Flutter/Kotlin Multiplatform/React Native

Группа рішень (мова + інфраструктура компіляції + набір бібліотек), орієнтованих на створення кросс-платформних додатків. Корисні у розробці додатків загального призначення, які не мають глибокої інтеграції з функціями платформи (або мають дуже обмежену), функціональність яких майже однакова на різних платформах, а зовнішній вигляд має бути або однаковим, або обмежено адаптованим під платформу (наприклад, мати нативний вигляд елементів керування (кнопок, полів вводу тощо) та нативні для платформи підходи до навігації). Дозволяють розробляти додатки під різні платформи (мобільні: iOS, Android; десктопні: Windows, Mac OS, Linux; а також під веб) використовуючи одну кодову базу, та відокремлючі код, який є специфічним для платформи, у окремі модулі.

У додатку, який є предметом цієї роботи, використання таких інструментів є недоцільним. По-перше, додаток не є кросплатформним, по-друге, більшість коду додатку є взаємодія з аудіо-системою пристрою (телефону), яка суттєво відрізняється на різних платформах. Інтерфейс користувача, у свою чергу, досить простий і становить меншу частину коду. По-третє, для успішної реалізації означеного додатку треба використовувати байндінги до специфічних АПІ пристрою, які, скоріш за все, не були написані раніше (або обмежені маленькою підмножиною самих популярних АПІ).

В принципі, є можливість створювати ці байндінги самому, але це досить великий обсяг коду і великий обсяг тестування.

Тому вважаю використання кросплатформених інструментів у даному випадку недоцільним.

Підхід, який використовує Web-view

Цей підхід використовує веб-браузер з прихованими елементами керування (так званий web-view), в який завантажується HTML сторінка та пов’язані ресурси (Javascript, CSS, зображення тощо). Ресурси можуть завантажуватися як локально (з пристрою), так і з мережі.

Цей підхід був досить популярним для додатків без глибокої інтеграції з платформою, тому що дозволяв запакувати існуючий веб-додаток у мобільний додаток з мінімальними змінами.

Наразі цей підхід є застарілим, і замість нього рекомендується використовувати Progressive Web Applications, які можуть розповсюджуватися без задіяння магазину додатків платформи.

У додатку, який є предметом цієї роботи, використання цього підходу є недоцільним з причин, аналогічних причинам для кросплатформених інструментів. Також написання бібліотек для доступу к АПІ платформи за допомогою джаваскрипта є досить складним.

2.2 Swift UI – сучасна бібліотека для побудування графічного інтерфейсу користувача за допомогою кода, рекомендована Apple.

SwiftUI - це декларативний фреймворк для розробки інтерфейсів користувача в програмах для платформ Apple, таких як iOS, macOS, watchOS та tvOS. Основою SwiftUI є ідея визначення стану та відображення інтерфейсу на основі цього стану.

Однією з ключових концепцій є "декларативність" - ви описуєте, як повинен виглядати ваш інтерфейс, а система автоматично стежить за змінами стану та оновлює інтерфейс відповідно. Це робить код більш зрозумілим і легше відлагоджуваним.

SwiftUI використовує властивості для визначення стану компонентів та об'єктів інтерфейсу. Зміни властивостей ведуть до автоматичного оновлення відповідних відображень. Ви також можете використовувати зв'язування даних для зручного зв'язування даних між різними частинами інтерфейсу.

SwiftUI також пропонує широкий набір вбудованих елементів управління, таких як кнопки, тексти, списки та інші. Компоненти можна гнучко налаштовувати та стилізувати. Важливою особливістю є можливість використовувати візуальні ефекти та анімації для створення динамічних і привабливих інтерфейсів.

SwiftUI також підтримує реактивне програмування, де зміни стану можуть автоматично викликати оновлення інтерфейсу. Це дозволяє створювати ефективні та відзивчиві додатки. Усе це робить SwiftUI потужним та зручним інструментом для розробки користувацьких інтерфейсів на платформах Apple.

Давайте розглянемо деякі інші ключові аспекти SwiftUI:

Stacks та Containers:

SwiftUI використовує контейнери, такі як HStack, VStack та ZStack, щоб організувати елементи інтерфейсу у горизонтальній, вертикальній або зазначеній орієнтації. Це дозволяє легко створювати складні макети.

Modifiers:

Ви можете використовувати модифікатори для зміни вигляду та поведінки елементів. Наприклад, .foregroundColor(), .font(), .padding() дозволяють вам налаштовувати властивості відображення.

Спостереження за Об'єктами:

SwiftUI автоматично визначає зміни властивостей та оновлює інтерфейс, але ви також можете використовувати @State, @Binding та інші властивості для спостереження за об'єктами та власноручно ініціювати оновлення.

Графічні Контексти:

SwiftUI надає контексти для малювання форм та графічних елементів, використовуючи Canvas та Path. Це дозволяє вам створювати власні візуальні компоненти.

Інтеграція з UIKit:

Ви можете використовувати SwiftUI разом із UIKit, інтегруючи його в існуючі додатки або використовуючи UIKit-елементи в SwiftUI, що дозволяє плавний перехід між старим та новим кодом.

Анімації:

SwiftUI спрощує створення анімацій за допомогою анімаційних модифікаторів та вбудованих ефектів. Вони можуть бути легко додані для поліпшення взаємодії та вигляду додатків.

Система Орієнтована на Проекти:

SwiftUI розроблено з урахуванням великих проектів, і його структура дозволяє легко організувати код та компоненти для зручної розробки та обслуговування.

SwiftUI представляє сучасний підхід до розробки користувацьких інтерфейсів, полегшуючи процес створення ефективних та красивих додатків для платформ Apple.

У дипломній роботі для побудови графічного інтерфейсу використана бібліотека Swift UI. Вона прийшла на зміну Interface Builder, який дозволяв будувати графічний інтерфейс за допомогою спеціального дизайнера

інтерфейсів, вбудованого у XCode. На відміну від Interface Builder, Swift UI використовує опис інтерфейсу за допомогою коду, і має наступні переваги:

- Опис інтерфейсу має зрозумілий синтаксис, який призначений для написання та читання людиною. Interface Builder має XML формат, який формується і читається машиною. Теоретично можливо змінювати файли Interface Builder вручну, але це досить складно і ненадійно.
- Оскільки опис інтерфейсу – це код, можна використовувати усі кращі практики розробки, як то: модулярізацю, абстракцію, перевикористання коду тощо.
- Опис інтерфейсу зручно зберігати у системах контроля версій, також зручно вирішувати конфлікти об’єднання при одночасній зміні різними користувачами

2.3 AudioKit

AudioKit це екосистема для роботи зі звуком для iOS. Вона є надбудовою над нативними АПІ платформи: AVFoundation та CoreXXX (CoreAudio, CoreMIDI тощо). Бібліотека надає потужний набір для роботи зі звуком, у тому числі:

- Конфігурація мікшера для вводу та виводу звука
- Конвертація форматів аудіо
- Робота з MIDI
- Додавання ефектів до звука у реальному часі, застосування фільтрів тощо
- Синтез (генерація) сигналу
- Набір контролів (елементів керування) для аудіододатків

У додатку, що є предметом цієї роботи, використані наступні можливості бібліотеки:

- Базова конфігурація пайплайну для відтворення аудіо
- Створення MIDI файлів у пам’яті

- Завантаження бібліотек звучання музичних інструментів у форматі SoundFont

- Програвання нот у реальному часі при натисканні клавіатури піаніно на екрані

2.4 Keyboard - Swift UI пакет, який дозволяє вивести на екран клавіатуру фортепіано

Пакет надає можливість вивести на екран інтерактивну клавіатуру фортепіано. Зокрема, доступні такі налаштування:

- Кількість клавіш, які відображаються на екрані
- Початкова клавіша, з якої починається клавіатура
- Зовнішній вигляд натиснутої клавіші
- Налаштування зовнішнього виду окремих клавіш (це використовується для відображення нижньої ноти інтервала, який був відтворений)
- Реакція на натискання та закінчення відпускання клавіші (використовується для програвання звуку певної клавіші)
-

2.5 SQLite.swift – пакет для роботи з СУБД SQLite

Додаток використовує SQLite для зберігання статистики вправ користувача.

SQLite – це система управління базами даних, яка є тою, що вбудовується у додаток. Тобто, додаток містить у собі весь необхідний код для роботи з базою, і не потребує встановлення додаткового ПЗ. База даних зберігається при цьому як набір файлів, у папці застосунка.

SQLite надає досить потужні функції, зокрема підтримується велика частина мови SQL, транзакції та ACID гарантії.

SQLite.swift – це бібліотека для роботи з SQLite. Підтримує усі парадигми Swift, у тому числі асинхронність.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ ВИПАДКОВИХ МУЗИЧНИХ ПОСЛІДОВНОСТЕЙ З РЕАЛІЗАЦІЄЮ НА БАЗІ SWIFT

3.1 Базовий опис функціональності додатку з точки зору користувача

Функціональність додатку з точки зору користувача полягає у програванні випадкового музичного інтервалу, після прослуховування користувач має обрати на фортепіанній клавіатурі, яка є на екрані.

Після програвання інтервалу нижня нота інтервала відображається на клавіатурі іншим кольором, користувач має обрати і натиснути правильну клавішу, яка відповідає верхній ноті зіграного інтервала.

Додаток має наступні екрани:

Стартовий екран.

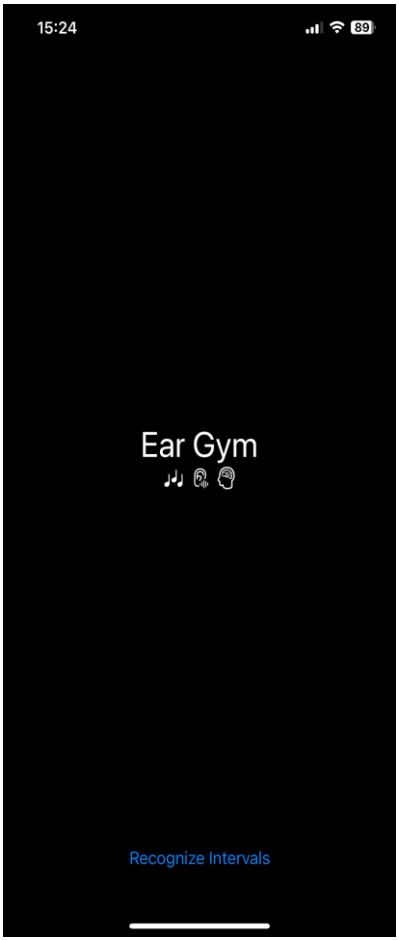


Рисунок 3.1 Стартовий екран додатку

Стартовий екран містить у собі список функцій додатку. На даний час додаток має єдину функцію, але з додаванням функціоналу цей екран буде доповнюватися.

Стартовий екран вправи “розрізнення інтервалів на слух”:

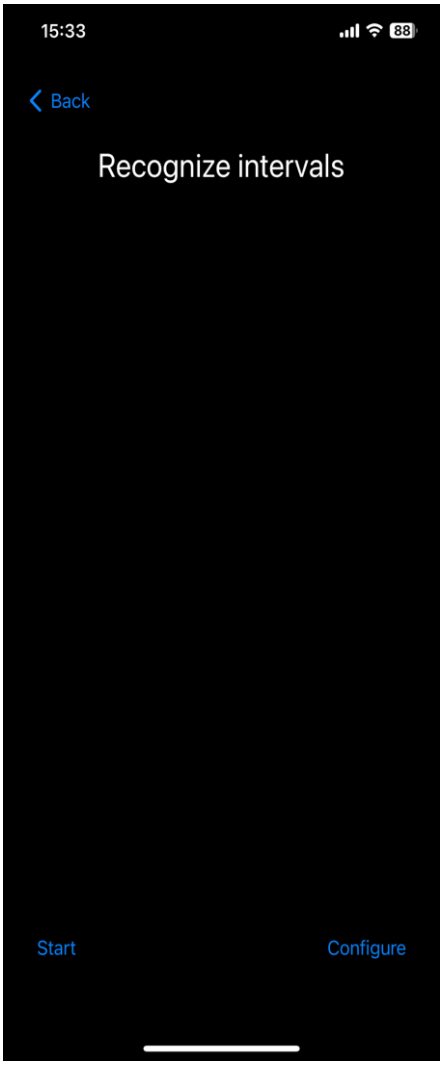


Рисунок 3.2 Стартовий екран вправи по розрізненню інтервалів на слух

З цього екрану можна почати вправу, або перейти до налаштувань.

Екран вправи “розрізнення інтервалів”, стан “Очікування відповіді користувача”

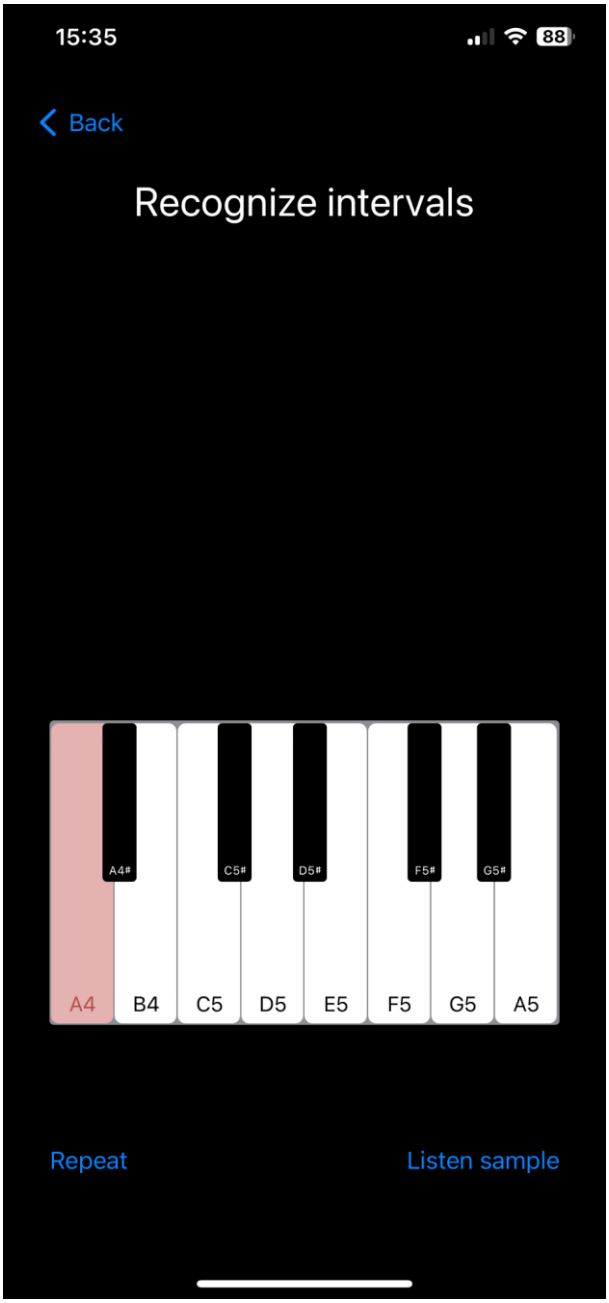


Рисунок 3.3 Очікування вибору інтервала

З цього екрану можна виконати наступні дії:

- Обрати другу ноту інтервалу шляхом натискання клавіші піаніно (перша (нижня) нота інтервала відмічена червоним кольором, у даному випадку це нота Ля першої октави). При натисканні клавіші буде зіграна нота, яка відповідає цій клавіші. Якщо ноту не відпускати, а переміщати палець на інші ноти, вони будуть зіграні, і можна “підібрати” інтервал на слух. Цей режим

можна вимкнути у налаштуваннях, бо він значно спрощує пошук правильної відповіді

- Програти інтервал ще раз – кнопка Repeat
- Прослухати зразки різних інтервалів – кнопка Listen Sample

Після натискання другої клавіші ми попадаємо на екран результату розрізнення:

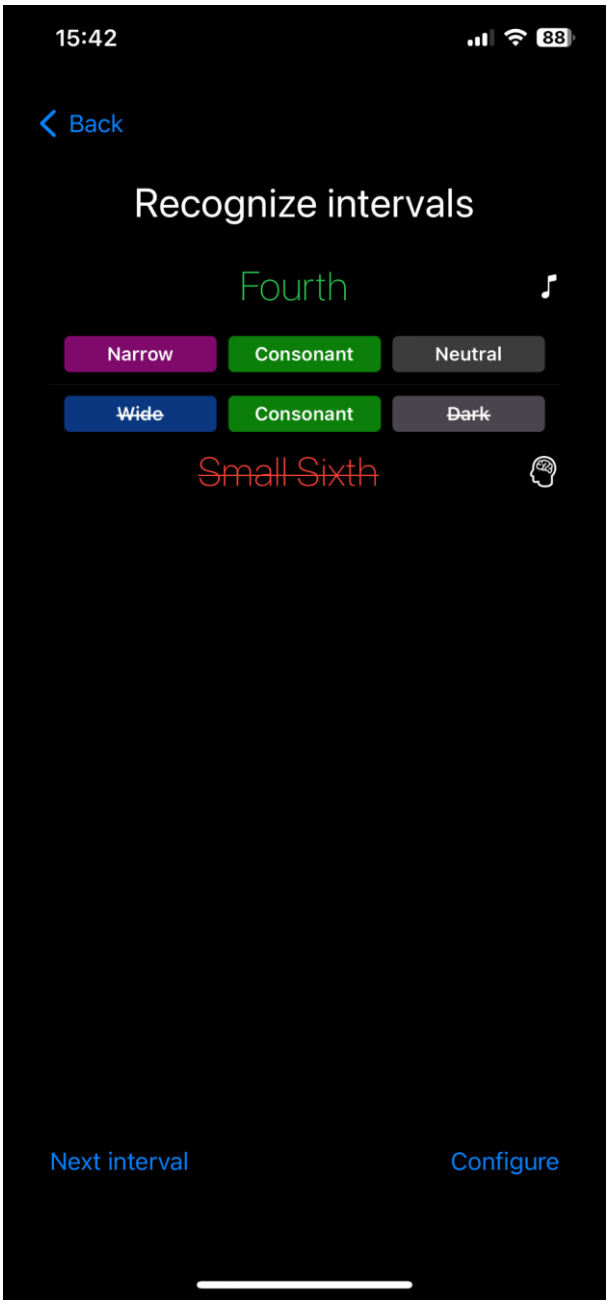


Рисунок 3.4 Результати розрізнення інтервалу

У даному випадку користувач не вгадав. Був програний інтервал “кварта”, а користувач обрав інтервал “мала секста”.

Зверху на екрані зеленим кольором відображена назва інтервалу, який був програний (він відмічений знаком ноти), знизу – той, який обрав користувач, зеленим або червоним кольором, в залежності від того, чи правильно був обраний інтервал. Обраний інтервал відмічений знаком розуму, що показує, що користувач мав обрати відповідь на основі роздумів.

На приведеному екрані є кнопки “Наступний інтервал”, яка програє наступний випадковий інтервал, а також кнопка яка відкриває екран налаштувань вправи, який детально буде розглянуто далі.

Цей екран також показує характеристики програного та обраного інтервалів.

Додаток використовує три характеристики для інтервалів.

- Інтервал широкий чи вузький. Широкий інтервал має більш ніж п’ять півтонів між верхньою чи нижньою нотою. Деякі інтервали звучать дуже схоже, і визначення відстані між нотами дозволяє розрізнити ці інтервали

- Консонантний чи дисонансний інтервал. Це характеристика звуку, який виходить при одночасному програванні нот інтервалу. Консонантне звучання є красивим, гармонійним, звуки наче зливаються один з одним. З фізичної точки зору частоти консонантних інтервалів співвідносяться між собою як невеликі цілі числа. Наприклад, 1:2 для октави, 2:3 для квінти і так далі. Дисонансні інтервали звучать негармонійно, дратівливо. Зазвичай дисонантні інтервали мають невелику кількість півтонів від нижньої до верхньої, або від верхньої до нижньої ноти, перенесеної на октаву вгору.

- Нейтральний, мажорний чи мінорний. Вважається, що інтервал може мати “забарвлення”, мажорне – світле, радісне, або мінорне – темне, сумне, або ж бути нейтральним.

Ці характеристики інтервалів наведені у додатку як допомога у розрізненні. Зазвичай, учнів вчать визначати наведені характеристики на слух, і на основі цього визначати вже програний інтервал.

Характеристики всіх інтервалів наведені у таблиці 3.1

Таблиця 3.1

Характеристики усіх інтервалів

Назва інтервалу	Кількість півтонів	Широкий чи вузький	Консонантний чи дисонансний	Забарвлення
Прима	0	Вузький	Сильно консонансний	Нейтральне
Мала секунда	1	Вузький	Сильно дисонансний	Нейтральне
Велика секунда	2	Вузький	Дисонансний	Нейтральне
Мала терція	3	Вузький	Слабо консонантний	Мінорне
Велика терція	4	Вузький	Слабо консонантний	Мажорне
Кварта	5	Вузький	Консонантний	Нейтральне
Тритон	6	Широкий	Слабо дисонансний	Нейтральне
Квінта	7	Широкий	Сильно консонантний	Нейтральне
Мала секста	8	Широкий	Слабо консонантний	Мінорне
Велика секста	9	Широкий	Слабо консонантний	Мажорне
Мала септіма	10	Широкий	Сильно дисонансний	Нейтральне
Велика септіма	11	Широкий	Сильно дисонансний	Нейтральне
Октава	12	Широкий	Сильно консонантний	Нейтральне

Інтервали можуть повторюватися через октаву (октави), деякі мають власні, але розрізнення таких інтервалів потрібно дуже рідко, і за характеристиками вони такі самі, як і інтервали у межах однієї октави, але ці характеристики виражені слабше. Додаток не містить функціоналу по тренуванню розрізнення інтервалів ширше за октаву.

Екран прослуховування зразків інтервалів

Рисунок 3.5 Прослуховування зразків інтервалів

Користувач може зайти у цей екран перед тим, як обирати ноту, і прослухати, як різні інтервали звучать на обраному музичному інструменті.

Екран налаштувань вправи

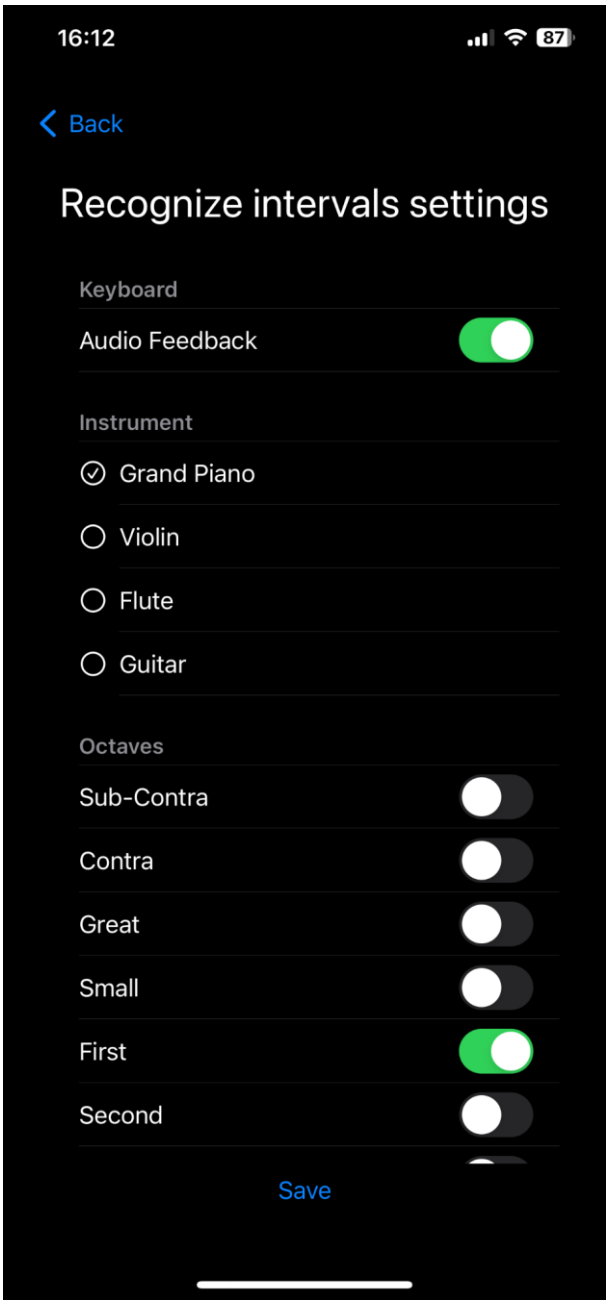


Рисунок 3.6 Екран налаштувань, частина перша

Екран налаштувань доволі великий, і буде описаний по частинам. На першій частині ми бачимо три секції:

- Audio Feedback – чи буде програватися звук при натисканні клавіші віртуального піаніно. Якщо це налаштування ввімкнено, то розрізнення стає набагато простішим, бо натискаючи та переміщуючи палець можна на слух

знайти правильний інтервал. Якщо ж це налаштування вимкнено, користувач має правильно вибрати інтервал з першого разу.

- Інструмент. Додаток дозволяє програвати інтервали на різних музичних інструментах. Наразі доступні фортепіано, скрипка, флейта та гітара. Наявність цього налаштування відрізняє додаток від існуючих аналогів, і дозволяє тренувати розрізнення на різних звуках. Зазвичай учень звикає до звуку одного інструмента (фортепіано у більшості випадків, або скрипки для скрипалів), і має складнощі при розрізненні інших інструментів.

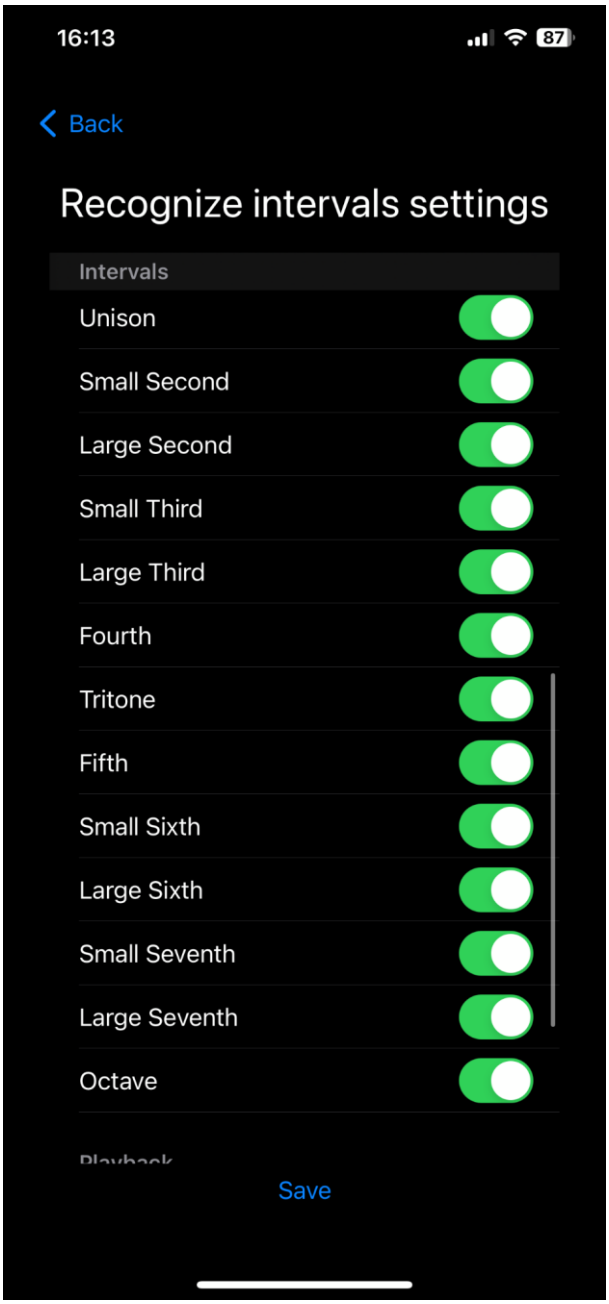


Рисунок 3.6 Налаштування вправи, частина друга

- Октави. У яких октавах можуть програватися інтервали. При навчанні сольфеджіо зазвичай використовуються лише октави, близькі до висоти розмовного голосу (мала, перша і друга), тому розрізнення дуже високих або дуже низьких нот може складати труднощі. Додаток дозволяє тренувати це.

На другій частині можна обрати, які саме види інтервалів будуть програватися. Це корисно, якщо є труднощі з розрізненням певних інтервалів. Зазвичай, це близькі за характеристиками, але різні за шириною інтервали, наприклад, секунди і септіми, або терції та сексти. За допомогою цих налаштувань можна цілеспрямовано тренуватися їх розрізняти.

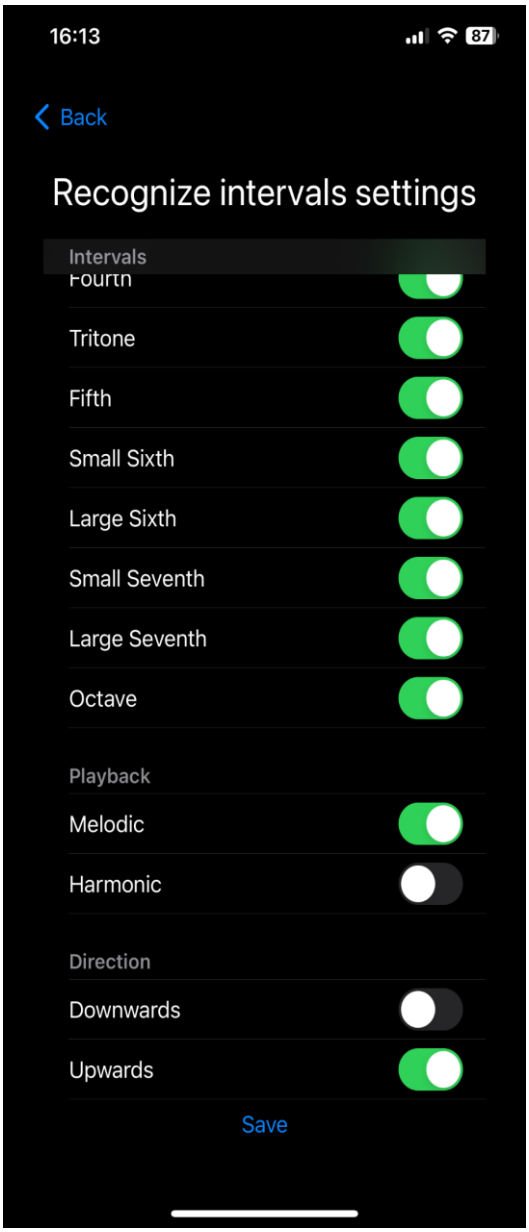


Рисунок 3.7 Налаштування вправи, частина третя

На третій частині екрану налаштувань ми бачимо наступні пункти:

- Playback – види відтворення інтервалів. Музичний інтервал – це дві ноти, і він може бути зіграний як дві ноти одночасно – так званий гармонічний інтервал, так і як дві ноти одна за іншою – так званий мелодійний інтервал. Група налаштувань дозволяє обрати види відтворення.

- Direction – це підгрупа, яка керує тим, як виконуються мелодійні інтервали. Downwards – спочатку відтворюється верхня нота, потім нижня, Upwards – спочатку нижня, потім верхня. Зазначимо, що мелодійне виконання з верхньої до нижньої ноти є самим складним для визначення інтервалу.

3.2 Архітектура додатку

Додаток написано на мові програмування Swift у IDE XCode.

Додаток складається з наступних модулів:

- AudioPlayer. Відповідає за конфігурацію аудіо пайплайну, завантаження Soundfonts файлу, створення MIDI файлів у рантаймі та їх програвання.

- Harmony. Реалізує базову музичну гармонію, зокрема побудову заданого інтервалу від певної ноти, визначення інтервала за двома нотами та інше.

- Controls. Ізольовані компоненти інтерфейса користувача. Наприклад, екранна клавіатура фортепіано.

- Pages. Сторінки (екрани) додатку

- DataAccess. Робота з базою даних для зберігання статистики. Схема даних, міграції, сутності, та клас репозиторію, який інкапсулює роботу з базою.

3.2.1 Архітектура роботи зі звуком

AudioEngine - обгортка над системним класом AVAudioEngine. Цей класс керує:

- графом аудіо-вузлів.
- відтворенням звука
- додаванням ефектів до звука

Аудіо вузли можуть бути:

- програвачами файлів різних форматів. Підтримується програвання одразу із файла, без завантаження файла у оперативну пам’ять

- мікшерами. Вузол мікшера об’єднує звук з декількох джерел, а також керує гучністю кожного джерела звука

- вузлами аудіо вводу. Дозволяє отримувати звук з мікрофона або з MIDI пристрою

- самплерами, дозволяє програвати MIDI файли або відтворювати звучання музичних інструментів у реальному часі при підключенні інструмента по протоколу MIDI. Дозволяє також завантажувати семпли, наприклад, у форматі SoundFonts2 та обирати інструменти, звучання якого буде використовуватися для відтворення

Також AudioEngine дозволяє обирати пристрій для виводу звука. Зазвичай, у програмах для iOS це не потрібно, і можна використовувати пристрій за замовчуванням. Пристрій виводу звука для iOS можна змінити через системні налаштування.

MIDISampler. Обгортка над системним класом AVAudioUnitSampler. Керує відтворенням MIDI файлів або програванням аудіо-даних, отриманих по протоколу MIDI, у реальному часі.

Дозволяє завантажити семпли зі зразками звучання музичних інструментів з файлів SoundFonts2. Також дозволяє обрати, який інструмент буде використано для відтворення.

AppleSequencer. Обгортка над системним MIDI секвенсором.

Секвенсор дозволяє будувати MIDI файли програмно, використовуючи для цього наданий API.

API секвенсора відображає базову структуру MIDI файла, яка складається з набора доріжок (tracks). Кожній доріжці може бути призначений музичний інструмент. Доріжка містить у собі набір нот. Кожна нота має наступні властивості:

- висоту ноти. Визначає, власне, яка саме нота має бути відтворена. Висота ноти задається числом, яке є порядковим номером ноти у послідовності нот з інтервалом у пів-тона. Нумерація починається з октави, нижчої на одну, ніж субконтроктава. Таким чином, до субконтроктави має номер 12, до контроктави – 24 і так далі.

- Момент часу, коли нота має бути відтворена. Моменти часу задаються у відліках (beats). MIDI файл має властивість “відліків за секунду”, що дозволяє керувати темпом відтворення без зміни тривалостей та моментів окремих нот

- Тривалість ноти. Задається також у відліках, відповідає за те, як довго нота буде звучати

- Сила натискання (velocity). Передає силу, з якою зіграна та чи інша нота, наприклад, з якою силою натиснута клавіша піаніно, чи як сильно притиснутий смичок до струни тощо. MIDI Velocity задається 8-бітним числом, тобто, сила натискання має 256 градацій.

Модуль Harmony

Модуль Harmony реалізує базові правила музичної теорії, які дозволяють

- Будувати заданий інтервал вгору від заданої ноти

- Аналізувати інтервал, отримуючи його характеристики (гармонійність, ширину та забарвлення)

- Визначати інтервал між двома нотами.

Модуль Harmony реалізує правила для європейський 12 ступінних ладів, що використовуються у європейській сучасній музиці.

Аналогічні правила з невеликими відмінностями також використовуються у стандарті MIDI.

Основні сутності у модулі Harmony:

MusicalNoteName – набір значень, (enum), який містить назви нот. У сучасній музичній освіті використовуються традиційні назви нот: до, ре, мі, фа,

соль, ля, сі. Але додаток використовує західну систему іменування нот, див. таблицю:

Таблиця 3.2 Назви нот українською та англомовні аналоги

Українська назва ноти	Західна назва
До	C
Ре	D
Мі	E
Фа	F
Соль	G
Ля	A
Сі	B
Сі-бемоль	H

Також MusicalNoteName містить метод для отримання кількості півтонів між нотою та нотою до (C у англомовній). Нота до є базовою (першою) нотою октави у стандарті MIDI, тож для спрощення ми використовуємо її також.

Octave – містить назви октав та їх номери.

Додаток використовує підмножину усіх існуючих октав, виключаючи найнижчі та найвищі. У сучасній музиці крайні октави в основній мелодії або в основному акомпанементі використовуються дуже рідко. Також їх відтворення потребує високоякісного обладнання.

Розрізнення інтервалів на слух не є дуже вимогливим до якості звука, але все ж буде надто ускладненим для наднизьких або надвисоких звуків.

У додатку доступні такі октави:

- Субконтроктава
- Контроктава
- Велика октава

- Мала октава
- Перша октава
- Друга октава
- Третя октава
- Четверта октава

Стандарт MIDI дозволяє відтворювати ноти на ще одну октаву вниз від субконтроктави.

Alteration - набір значень для знаків музичної альтерації.

У сучасній музичній теорії вирізняють чисті ноти (до, ре, мі, фа, соль, ля, сі) – на клавіатурі піаніно вони представлені білими клавішами, та альтеровані ноти.

Альтерована нота може бути:

- підвищена на півтона – позначається знаком ♯ (дієз)
- знижена на півтона – позначається знаком ♭ (бемоль)
- незмінна – позначається знаком ♮ (бекар). У додатку ця альтерація наразі не використовується, але додана для повноти.

Між двома чистими нотами може бути відстань або півтона (тоді застосування альтерації дасть іншу чисту ноту, наприклад E ♯ = F; або F ♭ = E), або два півтони. У цьому разі альтеровану ноту можна записати двома шляхами: як альтерацію вгору від нижчої ноти, або як альтерацію вниз від вищої, наприклад, C ♯ = D ♭).

У додатку, що є предметом цієї роботи, нота альтерується завжди з дієзом.

MusicalNote – структура, яка містить інформацію про конкретну ноту:

```
struct MusicalNote: CustomStringConvertible, Comparable {
    var name: MusicalNoteName
    var octave: Octave
    var alt: Alteration?
```

```
…
}
```

Як бачимо, нота визначається назвою, октавою та альтерацією.

Важливий метод для цієї структури є метод для визначення абсолютної позиції ноти:

```
func toAbsolutePosition() -> UInt8 {
    // C of SubContrOctave has number 12
    // C of ContrOctave has number 24 etc
    let octaveBase = Int8((octave.rawValue + 1) * 12)
    let altOffset = alt?.rawValue ?? 0
    return UInt8(octaveBase + Int8(name.semitonesFromC()) + altOffset)
  }
```

Цей метод визначає номер ноти відповідно до стандарту MIDI.

У стандарті міді нота – це число, яке визначає кількість півтонів даної ноти від до октави, нижчої від субконтроктави. Октава складається з 12 півтонів, тож До субконтроктави має номер 12, До контрокотави – 24 тощо.

3.3 Загальні відомості про MIDI та SoundFonts

Додаток використовує формат MIDI та формат банків звуків SoundFonts для створення та програвання довільних мелодій за допомогою звучання обраного музичного інструмента.

MIDI - це стандартний протокол для передачі музичної інформації між електронними музичними інструментами, комп'ютерами та іншими пристроями. MIDI файли не містять ніяких аудіоданих, натомість вони містять опис, як саме потрібний звук має бути зіграний.

На відміну від інших форматів це не оцифрований звук , а набори команд (ноти, що програються, посилання на інструменти, що програються, значення змінюваних параметрів звуку), які можуть відтворюватися по-різному

залежно від пристрою відтворення. Зручність формату MIDI як формату представлення даних дозволяє реалізовувати пристрої, що виробляють автоматичне аранжування за заданими акордами, а також програми 3D візуалізації звуку. Крім того, такі файли зазвичай мають на кілька порядків менший розмір, ніж оцифрований звук порівнянної якості.

MIDI-файл містить інформацію про:

- Ноти: Передає інформацію про ноти, їх довжину та інші параметри.
- Динаміка: Визначає силу звуку ноти.
- Контролери: Дозволяють контролювати параметри звучання, такі як вираз, реверберація, темп та інші.
- Програмні зміни: Встановлюють інструменти та звукові банки.

Як було зазначено, протокол MIDI використовується для взаємодії з різним обладнанням, не лише музичними інструментами, а й звуковими контролерами, а також освітлювальною технікою. Завдяки своїй простоті і компактності протокол забезпечує швидкість взаємодії з обладнанням, тому може використовуватися для виконання на інструменті у реальному часі майже без затримок, якщо комп’ютер достатньо потужний.

В багатьох сучасних цифрових інструментах є підтримка MIDI, або його варіації MIDI-over-USB, яка дозволяє під’єднувати обладнання стандартним кабелем USB.

Багато музичних інструментів, зокрема, цифрові піаніно та синтезатори, використовують MIDI в електронній схемі інструмента, що дозволяє обирати різні звучання, а, у моделях верхнього сегмента, також завантажувати свої банки звуків у сумісному форматі.

Файли MIDI можуть мати розширення .mid або .midi. Так як MIDI файли не містять аудіоданих, для відтворення необхідно якимось чином синтезувати звучання необхідного музичного інструмента.

Є два шляхи для цього:

- Програмні синтезатори. Використовуючи характеристики звучання певного музичного інструмента, та його фізичні характеристики, програмні

синтезатори генерують звук, поєднуючи сигнали різної форми та частоти, щоб імітувати основний тон, обертони, та темпоральні характеристики звука музичного інструмента (наростання звука при атакі, затухання, післязвучання тощо). Програмні синтезатори мають компактний розмір, бо не містять аудіоданих, але вимогливі до ресурсів центрального процесора, бо проводять досить складні обчислення у реальному часі. Для створення реалістичних звуків потрібно ускладнювати обчислення аж до фізичного моделювання коливань струн та корпусів інструментів. Тому цей підхід використовується рідко, частіше всього для вбудованих систем з малим об'ємом постійної пам'яті. Також досить часто базовий синтезатор поставляється з операційною системою, щоб мати змогу відтворювати MIDI файли, хай і з низькою якістю.

- Банки семплів. Альтернативою програмному синтезу є банки звуків. Це набір аудіо фрагментів, записаних з високою якістю з реальних музичних інструментів. Записуються декілька варіантів звучання (наприклад для піаніно: м'яка атака, сильна атака, стаккато (дуже різке, коротке натискання клавіші). Такий запис роблять для всіх нот, які можливо зіграти на інструменті. Після цього ці аудіосемпли програмно обробляються для того, щоб відтворити певну ноту: змінюється довжина згідно з довжиною ноти, яка потрібна, додаються ефекти, змінюється гучність тощо. Також для економії місця деякі проміжні ноти отримуються шляхом зміни висоти найближчої ноти з банка. Зміна висоти не дуже впливає на якість, якщо вона є невеликою (до 1-2 тонів). Банки семплів мають найвищу якість звучання, близьку до якості звучання фізичного інструмента. Але натомість вони займають багато місця на диску (зазвичай декілька гігабайт), та вимогливі до оперативної пам'яті (бо для швидкого відтворення у реальному часі аудіосемпли необхідно тримати в ОЗУ).

Sound Fonts – це формат зберігання банків звуків. Дозволяє зберігати необмежену кількість семплів, звучань різних музичних інструментів, та інше. Усе це зберігається як єдиний файл для зручності.

Sound font може містити:

- Зразки (samples): Зберігають записи реальних звуків інструментів.

- Параметри (parameters): Визначають характеристики звучання, такі як тембр, гучність, атака та інші.

- Петлі (loops): Дозволяють повторно відтворювати частини зразків для економії ресурсів.

Використання Sound Fonts:

Sound Fonts використовуються для покращення якості звуку в музичних програмах, секвенсорах та синтезаторах. Застосовуються для імітації різних інструментів та створення реалістичного звучання. Файли зазвичай мають розширення .sf2 (SoundFont 2).

Переваги формату MIDI:

- Компактний розмір
- Досить широкі можливості
- Легкий для програмної обробки: створення мелодій, аналізу, транспонування, тощо
- Досить висока якість звучання, за умови використання банків звуків
- Легко відтворити мелодію за допомогою різних музичних інструментів

Недоліки формату:

- Створений для запису та зберігання інструментальної музики. MIDI це набір нот + додаткова інформація, і він може лише озвучувати ноти. Наприклад, MIDI неможливо використовувати для запису вокалу, тому що він не містить аудіоданих.
- Потребує додаткового ПЗ для відтворення (синтезатора або банку семплів)

Для додатку, який є предметом цієї роботи, MIDI є ідеальним вибором, дозволяючи легко генерувати та відтворювати довільні мелодії.

3.4 Схема даних для зберігання статистики вправ

Додаток зберігає лог вправ. Лог зберігається локально на пристрої користувача. Для зберігання використовується база даних SQLite.

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Початковий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

База даних складається з наступних таблиць:

Таблиця 3.2 Схема таблиці sessions

Назва колонки	Тип	Опис
id	text	Первинний ключ таблиці. Містить унікальний ідентифікатор сесії у форматі GUID. Нова сесія створюється при кожному запуску додатку
started_at	text	Дата та час початку сесії у форматі ISO
ended_at	text	Дата та час закінчення сесії користування у форматі ISO

Таблиця 3.3 Схема таблиці recognize_intervals

Назва колонки	Тип	Опис
id	text	Первинний ключ таблиці у форматі GUID
session_id	text	Зовнішній ключ на таблицю sessions
played_at	text	Дата та час, коли інтервал було програно для користувача
recognized_at	text	Дата та час, коли користувач обрав відповідь. Ця та попередня колонки дозволяють визначити, як довго користувач думав до визначення.
first_note	int	Перша (нижня) нота зіграного інтервала
second_note	int	Друга (верхня) нота зіграного інтервала
playback_type	text	Тип відтворення інтервала. Може містити наступні значення: H - harmonic - обидві ноти відтворено одночасно MU - melodic up - спочатку була зіграна нижня нота, потім верхня MD - melodic down - спочатку була зіграна верхня нота, потім нижня
recognized_note	int	Нота, яку обрав користувач
played_interval	text	Інтервал, який був зіграний
recognized_interval	text	Інтервал, який обрав користувач
repeated_times	int	Значення, яке збільшується на 1 кожен раз, як

		користувач натискає кнопку Repeat на екрані розпізнавання інтервала.
playback_instrument	int	Ідентифікатор музичного інструмента, який було використано для відтворення
error_inverted	int (used as bool)	Містить значення 1, якщо користувач обрав обернений інтервал. Два інтервала є оберненими, якщо сума їх полутонів дорівнює 12 (цілій октаві)
error_wideness	int (used as bool)	Містить значення 1, якщо користувач помилився з шириною інтервала (наприклад, обрав октаву замість прими, або малу терцію замість малої сексти)
error_one_semitone	int (used as bool)	Містить значення 1, якщо користувач обрав інтервал, який відрізняється від зіграного на 1 півтон.

Основною таблицею є recognize_intervals. Новий запис додається кожен раз, коли користувач обирає відповідь на екрані Recognize Intervals. Використовуючи дані з таблиці, можна отримати відповіді на наступні запитання:

- Який відсоток правильних відповідей. Приклад SQL запиту:

```
with intervals as (select id, played_interval = recognized_interval as is_correct
                   from recognize_intervals)
select is_correct,
    count(id) count
from intervals
group by is_correct;
```

- Який інтервал частіше за все розпізнається помилково

```
-- What intervals have the most errors
with intervals as (select id, played_interval
                   from recognize_intervals
                   where played_interval <>
recognize_intervals.recognized_interval)
select played_interval,
    count(id) count
from intervals
group by played_interval
order by 2 desc;
```

- Який прогрес користувача у навчанні (як змінюється відсоток розпізнавання з часом)

```
-- Learning progress
with intervals as (select id,
                          datetime(played_at, 'unixepoch')
played_at,
                          datetime(played_at, 'unixepoch', 'start of day', 'weekday 0')
played_at_week,
                          played_interval = recognized_interval as is_correct
                   from recognize_intervals),
     weekly_stat as (select played_at_week,
                            is_correct,
                            count(id) as cnt
                     from intervals
                     group by played_at_week, is_correct)
select correct.played_at_week,
```

```
        correct.cnt - incorrect.cnt as diff
    from weekly_stat correct
        inner join weekly_stat incorrect on correct.played_at_week =
incorrect.played_at_week
    where correct.is_correct = true
     and incorrect.is_correct = false
    order by correct.played_at_week;
```

- Звучання якого музичного інструмента розпізнається краще, якого гірше
- Регулярність занять
- Тривалість занять

та інші.

3.5 Економічне обгрунтування

Додаток розроблений на запит музичної студії. Він надає учням можливість тренувати інтервальний музичний слух. Додаток має наступні особливості, які не надаються аналогами:

- Додаток безплатний
- Додаток орієнтований на єдину функцію – розрізнення музичних інтервалів. Учень може швидко перейти до вправи, не потрібно проходити уроки або курси. Теоретичні знання учень отримує від викладача на уроках.
- Додаток дозволяє відтворювати інтервали за допомогою різних музичних інструментів (фортепіано, гітара, скрипка, флейта), що сприяє навчанню і запобігає звиканню до звучання одного інструмента.
- Додаток відтворює інтервали у різних варіантах: гармонійному (обидві ноти звучать одночасно) і двох варіантах мелодійного (спочатку нижня нота, потім верхня, і навпаки).
- Можливість обирати октави, у яких будуть відтворюватися інтервали, а також відтворювати надширокі інтервали (через октаву)

- Додаток інтелектуально аналізує, які інтервали розрізняються краще, які гірше, і частіше програє ті інтервали, які користувач розрізняє гірше.

ВИСНОВКИ

У даній роботі проаналізовано сучасний стан ринку додатків для розвитку музичного слуха, досліджені їх функції, а також придатність для задачі розвитку музичного слуха у музикантів-аматорів.

У роботі запропоновано новий підхід до тренування розпізнавання музичних інтервалів на слух, зокрема, використовуючи статистику вправ, визначаються інтервали, які найгірше розрізняються користувачем, і такі інтервали пропонуються частіше, щоб закріпити більш складний матеріал.

Розроблений додаток також підтримує відтворення інтервалів за допомогою звучання різних музичних інструментів, для того, щоб навчити користувача відрізняти саме висотне співвідношення нот в музичному інтервалі, а не запам’ятовувати звучання на конкретному інструменті.

Розроблений додаток підтримує відтворення інтервалів у різних варіантах: одночасне звучання двох нот, послідовне від нижньої до верхньої, послідовне від верхньої до нижньої. Останній варіант відтворення викликає найбільші складності у учнів, і потребує окремого тренування.

У даній роботі досліджено сучасний стан розробки аудіо додатків на платформі iOS, вивчені існуючі інструменти та бібліотеки, та запропонована для використання бібліотека AudioKit.

Також досліджені сучасні засоби розробки інтерфейсу користувача для платформи iOS: Swift та SwiftUI.

Розроблено додаток для iPhone, який реалізує поставлені задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mobile operating systems' market share worldwide from January 2012 to October 2020. – Електрон. дан. – Режим доступу: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operatingsystems-since-2009/

2. Digital in 2020. – Електрон. дан. – Режим доступу: https://wearesocial.com/digital-2020

3. Офіційний сайт компанії Apple. – Електрон. дан. – Режим доступу: https://www.apple.com/ru/ios/what-is/

4. App Store. – Електрон. дан. – Режим доступу: https:// developer.apple.com/support/appstore/

5. iOS Human Interface. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/userexperience/conceptual/Mobile HIG/index.html

6. Офіційний сайт SQLite. – Електрон. дан. – Режим доступу: http://www.sqlite.org

7. В. Е. Дементьев. Информационно-вычислительные сети. – Ульяновск: УлГТУ, 2011. — 141с.

8. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol -- HTTP/1. – The Internet Society, 1999. – 114 с.

9. R.T.Fielding Architectural Styles and the Design of Network-based Software Architectures. – Hyderabad: University College of Science, 2000. – 180 с.

10. iOS Technology Overview. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneos techoverview/Introduction/Introduction.html

11. Cocoa Core Competencies. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/general/conceptual/devpediacocoacore/MVC.html 65

12. Steve Burbeck Applications Programming in Smalltalk-80TM: How to use Model- View-Controller (MVC) – ParcPlace Systems,1992. – 22 с.

13. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес, Приемы объектноориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.

14. Google Maps SDK for iOS. – Електрон. дан. – Режим доступу: https://developers.google.com/maps/documentation/ios/?hl=ru

15. Map Kit Framework Reference. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKit_Frame work_Reference/_index.html

16. Cocoa Application Competencies for iOS. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/ios/documentation/general/conceptual/DevpediaCocoaApp/Storyboard.html

17. Instruments User Guide. – Електрон. дан. – Режим доступу: https://developer.apple.com/library/mac/documentation/developertools/conceptual/instrumentsuserguide/Introduction/Introduction.html

18. App Store Resource Center. – Електрон. дан. – Режим доступу: https://developer.apple.com/appstore/index.html

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
Кафедра Інженерії програмного забезпечення автоматизованих систем

МАГІСТЕРСЬКА РОБОТА
на тему:
“Розробка моделі додатку для моделювання випадкових музичних послідовностей з реалізацією на базі Swift”

Виконав: студент групи ІСДМ-63
Сергій Токарєв

Керівник: д.т.н., професор
Андрій БОНДАРЧУК

Київ - 2023

Об`єкт дослідження - процес інтеграції технології IoT та AI.

Предмет дослідження – розробка додатку для роботи з аудіо для iOS за допомогою Swift

Мета роботи – дослідити функції, які надаються iOS для роботи зі звуком, а також побудову мобільного додатку за допомогою Swift UI

Завдання:

- проаналізувати сучасний стан ринку додатків для розвитку музичного слуху та навчання сольфеджіо;
- проаналізувати існуючі засоби та технології для розробки мобільних додатків;
- дослідити функції, які надає iOS для роботи зі звуком та розробки додатків, пов’язаних з аудіо обробкою;
- дослідити, як працювати з MIDI та SoundFonts;
- дослідити методи роботи з базами даних у iOS на прикладі бази SQLite;
- розробити додаток для тренування музичного слуха.

Вправи для розвитку музичного слуху

- *Визначення інтервалів на слух*
  - *Дуже просто і швидко, потребує інструмента, але може бути автоматизовано*
- *Спів інтервалів та більш складних послідовностей (гамм, тетрахордів, тощо)*
  - *Потребує тиші, приміщення, де можна співати, та контролю (викладача або розпізнавання висоти звуку)*
- *Музичні диктанти*
  - *Потребує часу на вправу, бажано мати пристрій з великим екраном та стілусом*
- *Визначення тональності на слух*
  - *Просто і швидко, але вправа є допоміжною і немає сенсу у систематичних тренуваннях*
- *Визначення гармонії на слух*
  - *Вправа просунутого рівня, досить складно для непрофесійних музикантів*
- *Визначення характеристик акордів на слух*
  - *Вправа для розвитку гармонійного слуху*
  - *Може бути додана як функція до розробленого додатку*

Вибір платформи для розробки додатку

- *Swift + SwiftUI*
  - *Рекомендована Apple платформа для розробки під iOS*
  - *Легкість інтеграції усіх існуючих API платформи (байндінги надаються Apple)*
  - *Сучасна мова програмування з підтримкою сучасних парадигм*
- *Objective C*
  - *Вважається застарілим з релізом Swift*
  - *Незручний спосіб опису інтерфейсів за допомогою Interface Builder*
- *Flutter/Kotlin Multiplatform/React Native*
  - *Орієнтовані на кросс-платформенні рішення*
  - *Легко розробляти інтерфейс і логіку*
  - *Складно використовувати платформенні API*
- *Web view based (Cordova etc)*
  - *Доцільно лише якщо вже є веб-додаток (і краще розглянути PWA)*
  - *Складний доступ до платформенних API*

Формат MIDI

MIDI file

Track 1

Note

Note

Note

Note

Note

...

Track N

Note

Note

Note

Note

Note

Note:
- Висота
- Час початку програвання
- Тривалість (час програвання)
- Сила натискання
- Інші атрибути (швидкість натискання, затримка та інше)

SoundFont file

Звучання піаніно

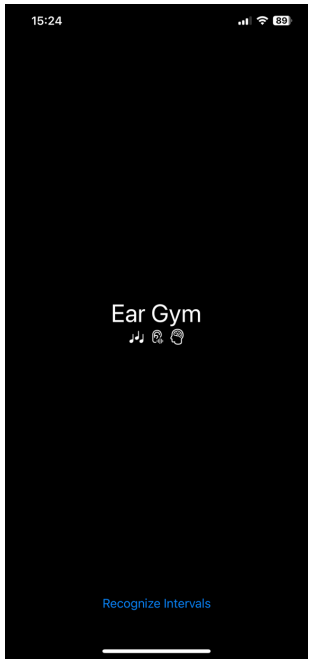
Звучання гітари

Звучання флейти

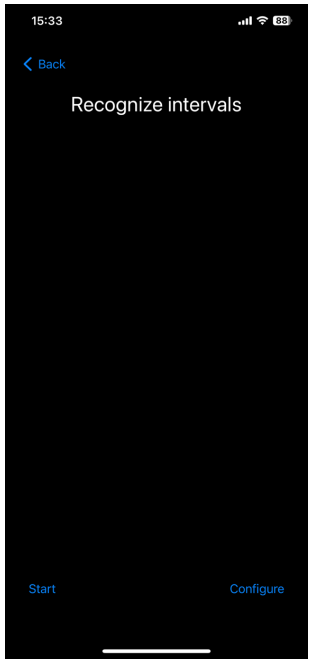
MIDI + Sound Font = Аудіо

Алгоритм генерації наступного інтервалу для розпізнавання

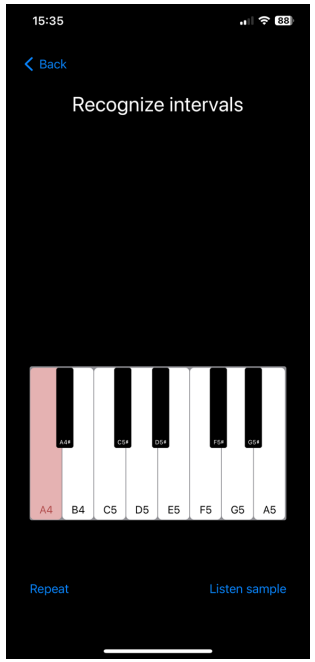
Основні екрани додатку



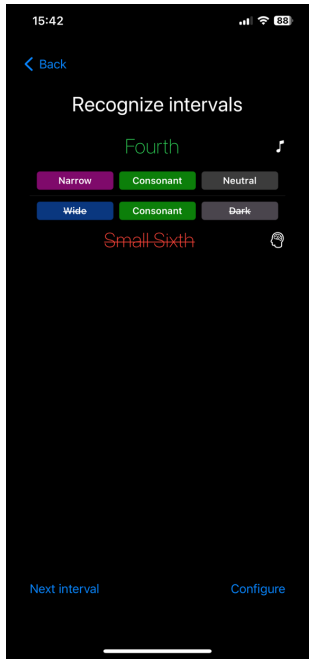
Початковий екран



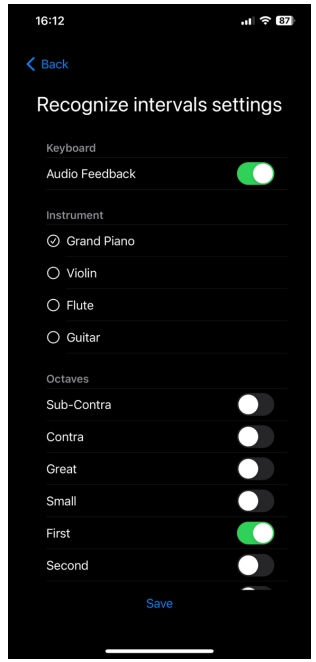
Початковий екран вправи



Мелодія зіграна, очікування відповіді



Результати розпізнавання



Налаштування вправи

ВИСНОВКИ

У даній роботі проаналізовано сучасний стан ринку додатків для розвитку музичного слуха, досліджені їх функції, а також придатність для задачі розвитку музичного слуха у музикантів-аматорів.

У роботі запропоновано новий підхід до тренування розпізнавання музичних інтервалів на слух, зокрема, використовуючи статистику вправ, визначаються інтервали, які найгірше розрізняються користувачем, і такі інтервали пропонуються частіше, щоб закріпити більш складний матеріал.

Розроблений додаток також підтримує відтворення інтервалів за допомогою звучання різних музичних інструментів, для того, щоб навчити користувача відрізняти саме висотне співвідношення нот в музичному інтервалі, а не запам’ятовувати звучання на конкретному інструменті.

Розроблений додаток підтримує відтворення інтервалів у різних варіантах: одночасне звучання двох нот, послідовне від нижньої до верхньої, послідовне від верхньої до нижньої. Останній варіант відтворення викликає найбільші складності у учнів, і потребує окремого тренування.

У даній роботі досліджено сучасний стан розробки аудіо додатків на платформі iOS, вивчені існуючі інструменти та бібліотеки, та запропонована для використання бібліотека AudioKit.

Також досліджені сучасні засоби розробки інтерфейсу користувача для платформи iOS: Swift та SwiftUI.