Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка додатку для навчання гри на музичних інструментах»

Виконав: студент 4 курсу
групи ЗПІ-21Б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Заболотчук А. В.
(прізвище та ініціали)
Керівник: д. ф., ст. викл. каф. ПЗ Стахов О. Я.
(прізвище та ініціали)
Рецензент: проф. каф. ОТ Захарченко С. М.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ О. Н. Романюк
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк _____
«21» березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Заболотчуку Артуру Віталійовичу

1. Тема роботи – розробка додатку для навчання гри на музичних інструментах.

Керівник роботи: Стахов Олексій Ярославович, доктор філософії PHD, старш. викл. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: графічний інтерфейс додатку – JavaFx; відтворення звуків та аудіофайлів – Java Sound API; вихідні дані для тестування – результати тестування додатку; мова програмування – Java; середовище розробки – IntelliJ IDEA; вихідні дані – результати тестування програми, скріншоти, результати виконання тестових сценаріїв.

4. Зміст текстової частини: аналіз сучасних методів оцінювання якості веб-навчання гри на музичних інструментах, порівняльний аналіз аналогів, аналіз методів розв'язання задачі, аналіз вхідних даних системи, розробка моделі системи, розробка алгоритмів роботи системи, варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення TuMusic, розробка модуля визначення характеристик завантаженої композиції, розробка модуля точного налаштування музичних параметрів, розробка модуля відтворення нот та акордів, розробка інтерфейсу програмного застосунку, тестування системи, розробка інструкції користувача, висновки, список використаних джерел, додатки.

5. Перелік ілюстративної частини: титульний аркуш, аналіз предметної галузі, особливості веб-навчання гри на музичних інструментах, мета дослідження, розробка моделі та алгоритмів програмного продукту, розробка програмних модулів додатку для навчання гри на музичних інструментах, тестування програми, наукова новизна та практичне значення.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Стахов О. Я., д. ф., ст. викл. каф. ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку технологій розвитку технологій веб-навчання гри на музичних інструментах та постановка задач	25.03.25 – 04.04.25	Вик
2	Розробка моделі та алгоритмів програмного продукту	04.04.25 – 27.04.25	Вик
3	Розробка програмних модулів додатку для навчання гри на музичних інструментах	27.04.25 – 12.05.25	Вик
4	Тестування програми	12.05.25 – 24.05.25	Вик
5	Оформлення матеріалів до захисту БКР	24.05.25 – 30.05.25	Вик

Студент _____ **А. В. Заболотчук**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **О. Я. Стахов**
(підпис) (ініціали та прізвище

АНОТАЦІЯ

УДК 00492

Заболотчук А. В. Розробка додатку для навчання гри на музичних інструментах : бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 52 с.

На укр. мові. Бібліограф.: 19 назв; рис.: 27; табл. 2.

У бакалаврській кваліфікаційній роботі проведено детальний аналіз сучасного вебнавчання гри на музичних інструментах. Також, було розглянуто основні принципи та проблеми у сфері навчання гри на музичних інструментах. Описано об’єкт, предмет, завдання, методи дослідження та мету дослідження, а саме покращення ефективності та результатів вебнавчання гри на музинчих інструментах.

Було розроблено програмні модулі для додатку, що призначені для підтримки навчального процесу гри на музичних інструментах. Реалізовано інструменти взаємодії з користувачем, включаючи відтворення навчального контенту, виконання інтерактивних завдань та створення зрозумілих і чітких налаштувань для музичних інструментів. Система дозволяє аналізувати дії користувача у реальному часі, оцінюючи точність і ритмічність виконання вправ. Розробка здійснювалася з використанням мови програмування Java, на платформі JavaFX, у середовищі IntelliJ IDEA. Використовувались такі бібліотеки такі як: Java Sound API і JFugue. Створені програмні рішення можуть бути використані у сфері дистанційного музичного навчання, індивідуальної практики та інтерактивних освітніх платформ.

Ключові слова: вебнавчання, музичні інструменти, додаток, Java Sound API, JFugue.

ABSTRACT

UDC 00492

Zabolotchuk A. V. Development of an application for teaching musical instruments: bachelor's thesis in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2025. 52 p.

In Ukrainian. Bibliography: 19 titles; Figures: 27; Table 2.

In the bachelor's thesis, a detailed analysis of modern web-based learning to play musical instruments was conducted. Also, the basic principles and problems in the field of learning to play musical instruments were considered. The object, subject, tasks, research methods and research goal, namely improving the effectiveness and results of web-based learning to play musical instruments, are described.

Software modules for the application were developed to support the learning process of playing musical instruments. Tools for user interaction have been implemented, including playback of educational content, completion of interactive tasks, and creation of clear and precise settings for musical instruments. The system allows you to analyze user actions in real time, assessing the accuracy and rhythmicity of the exercises. The development was carried out using the Java programming language, on the JavaFX platform, in the IntelliJ IDEA environment. The following libraries were used: Java Sound API and JFugue. The created software solutions can be used in the field of distance music education, individual practice and interactive educational platforms.

Keywords: web-based learning, musical instruments, application, Java Sound API, JFugue.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі дедалі більшої популярності набувають інтерактивні освітні платформи, зокрема ті, що орієнтовані на розвиток творчих навичок – таких як гра на музичних інструментах. Ефективність навчання у таких системах значною мірою залежить від інтерактивності, візуалізації, зручності подачі матеріалу, а також наявності механізмів зворотного зв’язку, які дають змогу аналізувати успішність виконання навчальних завдань.

Однією з актуальних задач є створення програмної системи, яка поєднує мультимедійні компоненти (аудіо, ноти, відео) з алгоритмами автоматичного аналізу дій користувача [1] для підтримки процесу самостійного навчання. Зокрема, йдеться про розпізнавання та оцінку правильності виконання вправ на музичному інструменті [2], точність введення нот, ритм, швидкість виконання тощо. Для досягнення цього необхідно поєднати аналітичні методи обробки навчальної інформації з елементами інтерфейсної взаємодії [3], що забезпечують зручність у використанні навіть для початківців.

У цьому контексті особливо важливо розробити програму, яка дозволить не лише відображати навчальні матеріали, але й аналізувати дії користувача в режимі реального часу, надаючи підказки та оцінки, спрямовані на підвищення ефективності навчального процесу. Актуальність теми пояснюється зростаючим попитом на дистанційні форми навчання, зокрема у сфері мистецтв, а також потребою у програмних рішеннях, які здатні підтримувати індивідуалізований підхід до навчання.

Метою бакалаврської кваліфікаційної роботи є створення програмних модулів для додатку, що сприятимуть підвищенню ефективності навчанню гри на музичних інструментах завдяки інтерактивному контенту, можливості точних налаштувань музичних параметрів та чітких позначень для гітарних технік.

Для досягнення цієї мети необхідно виконати такі завдання:

- визначити вимоги до функціональності та інтерфейсу додатку з урахуванням потреб користувачів-початківців;
- розробити алгоритми аналізу дій користувача;
- реалізувати програмні модулі для представлення навчального матеріалу;
- створити зручний, адаптивний графічний інтерфейс із можливістю використання на персональному комп'ютері;
- забезпечити можливість точної настройки музичних параметрів.

Об'єктом дослідження є процес дистанційного навчання гри на музичних інструментах з використанням вебтехнологій.

Предметом дослідження є методи розробки інтерфейсів, алгоритмів аналізу дій користувача та засоби інтеграції навчального контенту в освітні вебплатформи.

Практичне значення полягає у створенні універсального інструменту для підтримки індивідуального музичного навчання, який може бути використаний у музичних школах, онлайн-курсах, а також для самостійної практики. Запропоновану систему можна адаптувати для різних музичних інструментів, рівнів підготовки та інтегрувати в існуючі освітні платформи.

Методи дослідження. У процесі дослідження теми вебнавчання гри на музичних інструментах, застосовувались такі методи: аналітичні методи порівняння музичних додатків, метод поступового ускладнення завдань, алгоритми цифрової обробки звуку, методи оцінки ефективності навчання.

Наукова новизна отриманих результатів:

1. Запропоновано метод навчання гри на музичних інструментах з використанням технології відтворення аудіофайлів та звуків за допомогою бібліотек Java Sound API та JFugue. Дане поєднання покращує як точне та безперервчасте відтворення звуків, так і краще розуміння ритму у користувачів.
2. Модифіковано метод отримання результатів у ході навчання користувача, що відрізняється кращим представленням результатів, про ритмічність, помилки та попадання в ноти.

У першому розділі бакалаврської кваліфікаційної роботи було проаналізовано сучасний стан розвитку вебтехнологій у сфері дистанційного навчання музики, розглянуто основні підходи до побудови освітніх платформ, а також визначено вимоги до функціональності систем для навчання гри на музичних інструментах.

У другому розділі було розроблено концептуальну модель роботи програми навчання, визначено ключові програмні модулі, описано сценарії взаємодії користувача з інтерфейсом та обґрунтовано доцільність використання інтерактивного контенту.

У третьому розділі здійснено розробку програмних модулів системи: реалізовано навчальний інтерфейс, модулі взаємодії з нотним контентом, систему зворотного зв’язку та базові алгоритми оцінювання правильності виконання навчальних завдань. Також обґрунтовано вибір технологій розробки, зокрема використання мови програмуввання у середовищі IntelliJ IDEA.

У четвертому розділі проведено тестування функціональних можливостей розроблених модулів, оцінено їхню ефективність у процесі навчання та проаналізовано відгуки користувачів.

Даний застосунок розроблено для навчання музиці, з акцентом на гітарні техніки, і поєднує в собі функції редактора композицій, тренажера та метронома. Інтерфейс дозволяє детально працювати з нюансами виконання, що робить її корисним інструментом як для початківців, так і для досвідчених музикантів.

Таким чином, реалізація даного додатку сприяє підвищенню якості дистанційного навчання гри на музичних інструментах завдяки інтерактивному контенту, можливості точної настройки музичних параметрів та чітких позначень для гітарних технік.

Основні результати досліджень опубліковано у матеріалах конференції [4].

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ ВЕБ-НАВЧАННЯ ГРИ НА МУЗИЧНИХ ІНСТРУМЕНТАХ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз сучасних методів оцінювання якості веб-навчання гни на музичних інструментах

У сучасному цифровому середовищі стрімко зростає попит на інноваційні засоби дистанційного навчання, зокрема у сфері музичної освіти. З розвитком вебтехнологій та широким впровадженням мультимедійних платформ усе більшої популярності набувають онлайн-системи для навчання гри на музичних інструментах. Такі системи є важливими не лише для самостійного опанування гри, а й для підтримки формальної музичної освіти, особливо в умовах обмеженого доступу до традиційного викладання або індивідуального навчання.

Існуючі рішення у цій галузі здебільшого орієнтовані на відеоуроки, нотні тренажери, візуалізацію акордів, а також використання MIDI-клавіатур або мікрофонного вводу для аналізу гри користувача. Проте значна частина таких систем або не забезпечує якісний зворотний зв’язок, або обмежена функціональністю. Саме тому актуальною є розробка інтерактивних вебдодатків, що поєднують навчальні матеріали, мультимедіа і функції оцінювання гри користувача на основі аналітичних та експертних підходів.

Сучасні технології значно трансформували процес навчання грі на музичних інструментах, роблячи його більш інтерактивним, доступним та ефективним. Ось деякі з найновітніших підходів у вебнавчанні музики:

AI-Підтримка в Реальному Часі. Інтерактивний аналіз гри: платформи використовують штучний інтелект для аналізу звуку з мікрофона користувача та надають миттєвий зворотний зв’язок щодо точності нот, ритму та техніки. AI генерує персоналізовані вправи на основі прогресу учня [5].

Віртуальні та Доповнені Реальності (VR/AR). VR-симулятори: додатки імітують гру в оркестрі або на сцені, що допомагає відпрацювати навички виступу.

AR-додатки: проектують аплікатуру акордів прямо на гриф у реальному часі через камеру смартфона [6].

Гейміфікація. Платформи, перетворюють навчання на гру з рівнями, досягненнями та інтерактивними челенджами, що підвищує мотивацію [7].

Персоналізовані Курси на основі Big Data. Аналіз даних про стиль гри допомагає створити індивідуальний план тренувань, фокусуючись на слабких місцях [8].

Додаткові інструменти – такі як визначення темпу, виявлення помилок у нотах, візуалізація акордів та шкал, реєстрація успішності навчання – суттєво підвищують ефективність взаємодії учня із системою. Важливими є також адаптивні інтерфейси, що підлаштовуються під рівень користувача, а також кросплатформеність, яка дозволяє працювати з системою на будь-якому пристрої.

Таким чином, розробка вебзастосунку для навчання гри на музичних інструментах, що поєднує сучасні засоби візуалізації, аудіоаналізу та інтерактивного зворотного зв’язку, є актуальним і перспективним напрямом дослідження. Такий застосунок не лише підвищить доступність музичної освіти, а й забезпечить гнучкість та індивідуалізацію навчального процесу, сприяючи гармонійному розвитку музичних навичок користувачів різного рівня підготовки.

1.2 Порівняльний аналіз аналогів

Під час розробки програмного модуля для додатку з вивчення гри на музичних інструментах було здійснено аналіз існуючих рішень, які реалізують подібну функціональність. Це дало змогу визначити ключові переваги та недоліки наявних платформ, що використовуються для дистанційного музичного навчання, і на цій основі сформулювати вимоги до власного модуля. Аналіз проведено на прикладі трьох популярних систем, такі як: Fender Play, Jamorama, TrueFire. Перераховані застосунки є ідеальним порівнянням інтерактивного навчання, надання зворотного зв’язку та оцінювання прогресу користувача.

Fender Play. Розроблений переважно для початківців, цей додаток для навчання гри на гітарі пропонує різноманітні уроки, пристосовані для задоволення потреб практично кожного [9]. У Fender Play представлений широкий спектр стислих уроків для просунутих гітаристів. Ці уроки різняться за тривалістю: деякі тривають лише три хвилини, а інші є частиною більш об'ємних курсів, залежно від теми, що розглядається (див. рисунок 1.1).

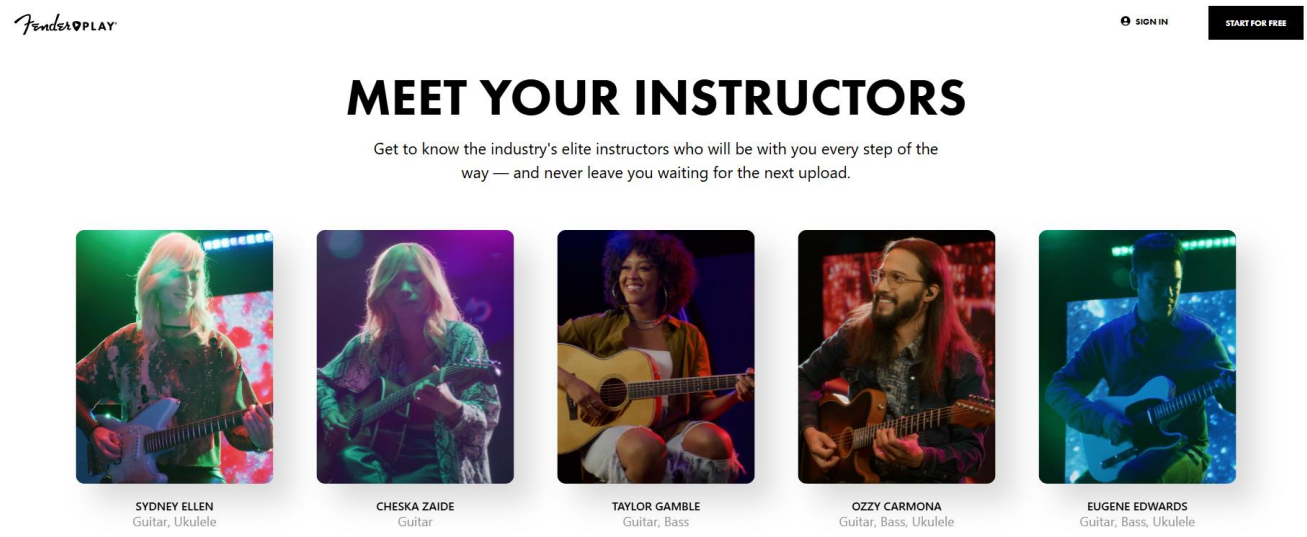

Рисунок 1.1 – Офіційний сайт Fender Play

Загалом, компактний формат робить навчання гри на гітарі більш легким, позиціонуючи Fender Play як одну з найкращих програм для навчання гри на гітарі.

Як і його конкуренти, Fender Play пропонує безкоштовну пробну версію, яка дозволить вам випробувати його численні функції та переваги, перш ніж оформити підписку.

Jamorama. Це не лише чудовий навчальний додаток, але й чудовий варіант для тих, хто трохи втомився від прив'язки до однієї програми [10]. Jamorama охоплює широкий спектр важливих гітарних навичок на своїх уроках, включаючи комплексні уроки з багатьох тем (див. рисунок 1.2); однією з важливих переваг Jamorama є довічний доступ до свого контенту, що відрізняє її від конкурентів, багато з яких не пропонують такої можливості.

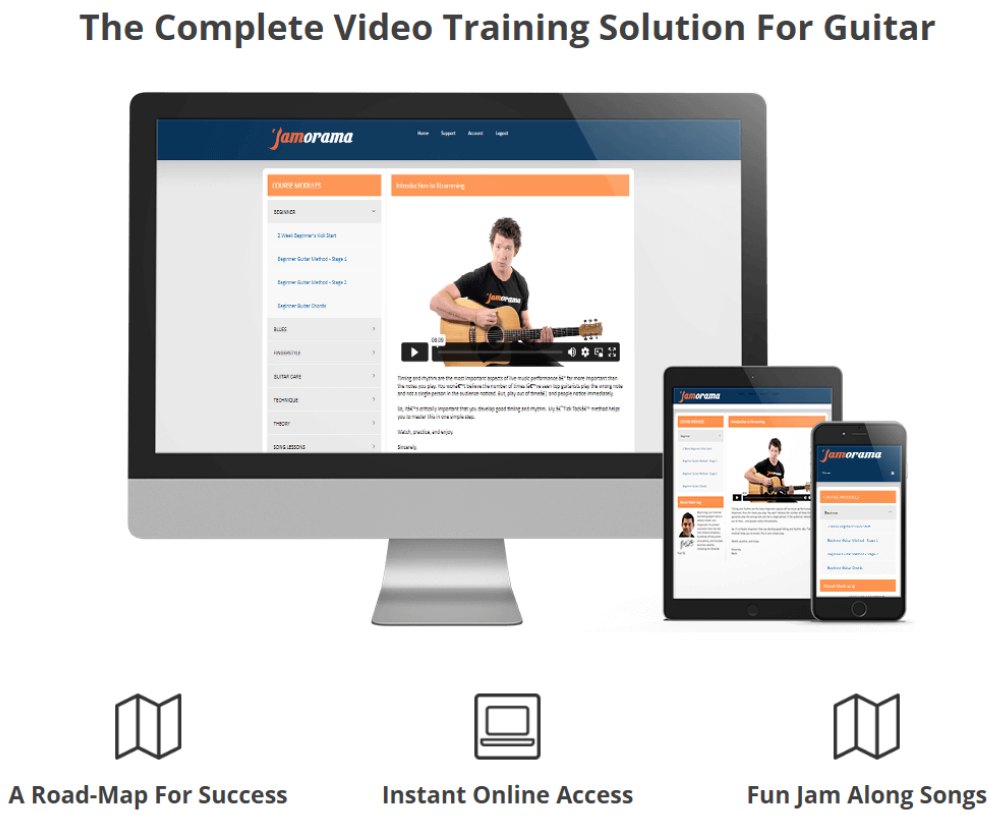

Рисунок 1.2 – Використання різних платформ у навчанні з Jamorama

Система також пропонує відмінне відстеження прогресу і веселі джем-сейшени, щоб зробити практику більш захоплюючою.

TrueFire. Даний аналог вирізняється тим, що пропонує можливість завантажувати окремі курси за одноразову плату, що є основною перевагою, яку не часто можна зустріти в інших програмах [11]. Завдяки цій унікальній функції доступу ви отримуєте більшу гнучкість і зручність. Truefire також позиціонує себе як універсальну платформу для гітаристів усіх рівнів майстерності. У той час як деякі платформи орієнтовані виключно на початківців або просунутих гравців, TrueFire досягає гарного балансу, пропонуючи щось для кожного.

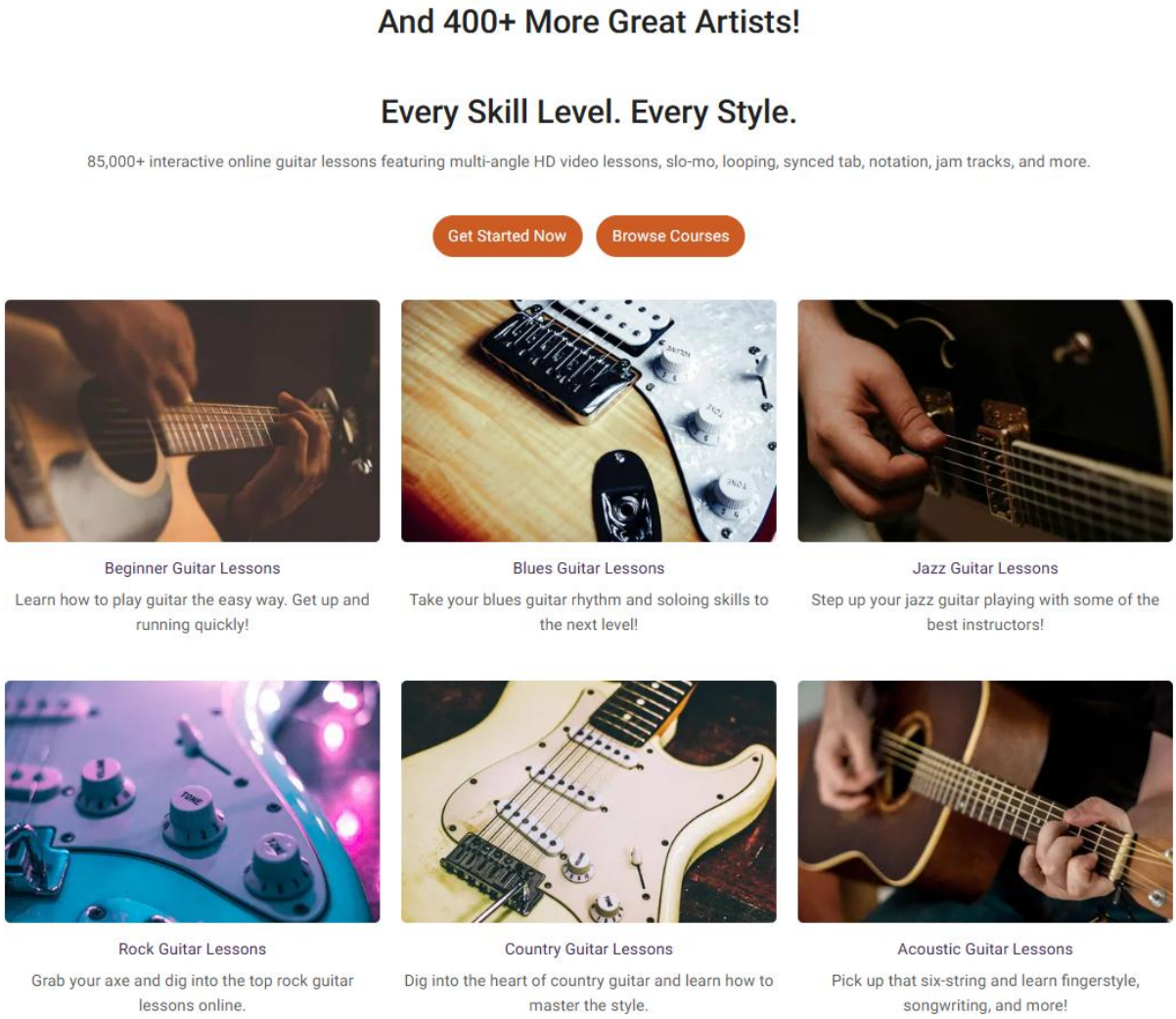

Рисунок 1.3 – Уроки для навчання гри на гітарі в TrueFire

Після аналізу аналога було створено таблицю порівняльних характеристик для оцінки за ключовими критеріями:

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Fender Play	Jamorama	TrueFire	Власний застосунок
1	2	3	4	5
Інтеграція у веб-застосунок	+	+	-	+
Функція контролю прогресу	-	+	-	+

Продовження таблиці 1.1

1	2	3	4	5
Наявність курсів	+	+	-	±
Простота інтеграції у React	+	±	-	+
Гнучкість налаштувань	+	-	+	+
Вартість (безкоштовність)	+	-	+	+
Загальна оцінка (%)	67%	53%	42%	79%

Таким чином, аналіз аналогічних продуктів дозволив визначити, що існуючі рішення мають як переваги, так і недоліки. Власний модуль поєднує інтеграцію у додатку, функцію контролю прогресу. Модуль було розроблено на Java з використанням новітніх технологій для інтеграції у додатку. У результаті було створено інтуїтивно зрозумілий інструмент аналізу зображень, який можна використовувати в рідному середовищі без необхідності підключення до сторонніх сервісів.

1.3 Аналіз методів розв’язання задачі

При розробці програмного модуля для навчання гри на музичних інструментах було проведено детальний аналіз існуючих методів вирішення цієї проблеми та інтегровано ефективне рішення в розроблену програму. Модуль базується на інтеграції у вебзастосунок, якісного та цікавого навчання на музичних інструментах.

Методи, реалізовані в програмному модулі, ґрунтуються на поєднанні алгоритмічних і педагогічних підходів, що дозволяють забезпечити відстеження навчального прогресу користувача. Зокрема, були реалізовані алгоритми автоматизованого аналізу виконання музичних вправ на основі таких показників,

як точність натискання нот, ритмічність та швидкість. Застосування цих методів дало змогу надати кількісну оцінку рівня володіння музичним матеріалом.

Програмний модуль створено у середовищі IntelliJ IDEA із використанням мови програмування Java. Його архітектура побудована так, щоб забезпечити адаптивність інтерфейсу, інтерактивну взаємодію з користувачем і підтримку різних музичних інструментів.

У процесі дослідження модуль було протестовано на реальних прикладах музичних занять для перевірки його функціональності, ефективності та відповідності вимогам до навчального програмного забезпечення. Результатом розробки є гнучка, масштабована та інтерактивна система, яка може бути використана в індивідуальному навчанні, музичних школах і онлайн-курсах.

1.4 Висновки

У проведеному аналізі розглядаються сучасні методи навчання гри на музичних інструментах. Було розглянуто сучасні види навчання, які підвищують якість, зручність і цікавість до гри на музичних інструментах. Також було наведено іноваційні методи навчання, які реалізуються на різноманітних пристроях, а також і за участі штучного інтелекту.

У порівняльному аналізі розглядаються Fender Play, Jamorama та TrueFire. Дані аналоги є чудовим прикладом програм для навчання гри на музичних інструментах, які мають як і плюси, так і мінуси. Була підготовлена порівняльна таблиця, яка показує, що власний модуль перевершує аналогічні продукти з точки зору гнучкості, простоти інтеграції та незалежності від сторонніх сервісів.

На основі проведеного аналізу за допомогою комбінації експертних оцінок та методів аналізу було сформульовано вимоги до кастомного модуля, який забезпечує точність та гнучкість оцінювання, легко інтегрується в Java-застосунок та є незалежним від зовнішніх сервісів.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

У процесі розробки програмних модулів до додатку для навчання гри на музичних інструментах застосовувалось середовище розробки IntelliJ IDEA, а основним інструментом для створення інтерфейсу користувача стала мова програмування Java [12]. Розробка передбачала використання сучасних фронтенд-інструментів, таких як Java Sound API для роботи з аудіовхідом/виходом, MIDI-пристроями, запису та відтворення звуку, графічний інтерфейс (GUI) – сучасний UI з анімаціями, підтримкою CSS, FXML для розмітки [13].

На першому етапі було розроблено модуль для завантаження та відтворення музичних файлів, який підтримує формати MP3, WAV та MIDI. Для відтворення аудіо – Java Sound API.

Для оптимізації роботи з даними було застосовано:

- Tritonus plugins (кешування аудіозаписів та результатів аналізу для швидкого доступу);
- JFrame (централізоване управління станом додатку).

Окремо було розроблено модуль, який забезпечує інструментарій для налаштування тривалості нот, динаміки та спецефектів у процесі створення музичних композицій.

Для зберігання та обробки даних використовувалась локальна база IndexedDB, що дозволяла кешувати результати обчислень та уникнути повторного аналізу при повторному завантаженні. Крім того, було застосовано JFrame для управління станом додатку, що дало змогу оптимізувати роботу з великими обсягами даних.

Останнім етапом стало тестування програмних модулів, яке включало перевірку точності розрахунків та продуктивності системи. Для написання юніт-

тестів було використано JUnit 5 (Unit Testing) та TestFX, а для оцінки продуктивності вебзастосунку – Lighthouse.

Отже, після опису розроблюваних етапів вебзастосунок поєднує в собі цікавий та інтерактивний функцонал, використовуючи сучасний стек. Система є масштабованою та може бути інтегрована з зовнішніми сервісами. Ключова перевага – гнучкий підхід до навчання, де алгоритми доповнюють людську експертизу.

2.2 Розробка моделі системи

З метою глибшого розуміння функціоналу програмних модулів вебзастосунку для навчання гри на музичних інструментах було побудовано модель роботи системи у вигляді діаграми діяльності UML (див. рисунок 2.1). Згідно з цією діаграмою, користувач для початку роботи із застосунком повинен завантажити музичні композиції.

У разі, якщо завантажений файл має невідповідний формат або пошкодження, система повідомляє користувача про помилку, запобігаючи подальшій обробці. Після успішного завантаження композиції, система переходить до етапу попередньої обробки даних, під час якого здійснюється перевірка правильності структури нот, тривалостей, темпу та інших параметрів.

У разі виявлення значних відхилень або помилок у музичному матеріалі, застосунок попереджає користувача, вказуючи, які саме елементи потребують корекції. Якщо ж композиція успішно проходить перевірку, система створює відповідні ноти та акорди до музичної композиції.

Результати аналізу відображаються в інтерфейсі у вигляді вікна, де можна взаємодіяти з композицією. Користувач має змогу взаємодіяти з навчальним матеріалом: переглядати завантажені музичні композиції, прослуховувати фрагменти та налаштовувати параметри аналізу відповідно до рівня підготовки. Такий підхід дозволяє адаптувати процес навчання під потреби кожного учня та забезпечити високу якість інтерактивного музичного навчання.

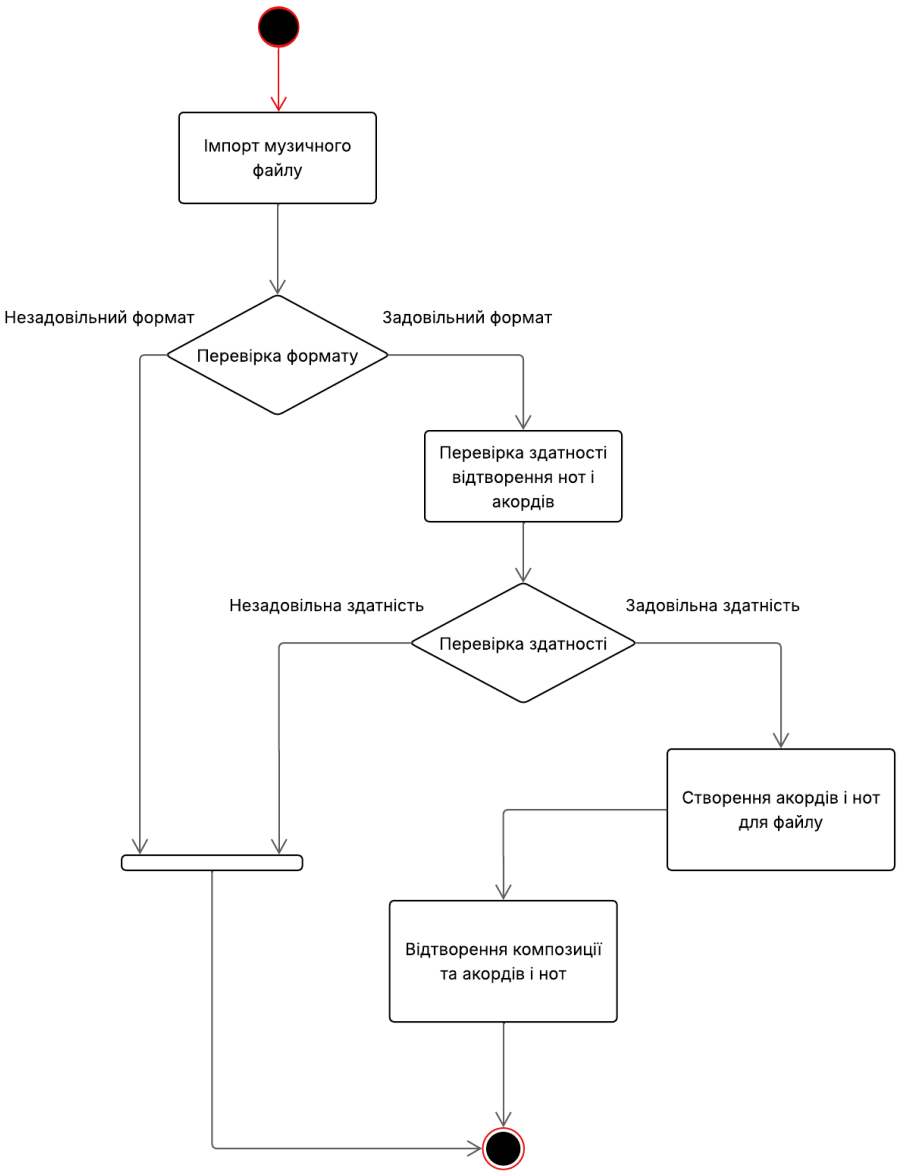

Рисунок 2.1 – Діаграма діяльності системи

Всі етапи роботи були спроектовані з акцентом на зручності та інтуїтивно зрозумілій взаємодії користувача з програмою.

Розробка цієї діаграми діяльності відбувалася в середовищі lucid.app.

2.3 Розробка алгоритмів роботи системи

Розробка програмних модулів програми для навчання гри на музичних інструментах відбувалась із використанням сучасних технік веброзробки.

Основним середовищем розробки було обрано IntelliJ IDEA, що надало змогу ефективно структурувати роботу, використовувати систему контролю версій і легко інтегрувати інструменти для тестування та налагодження. Для створення клієнтської частини використовувався бібліотека Junit 5, який забезпечує зручне оновлення інтерфейсу в режимі реального часу. Керування станом додатка здійснювалося за допомогою бібліотеки TestFX, що дозволяло підтримувати цілісність і узгодженість даних під час роботи з програмою.

Згідно із загальним алгоритмом роботи системи (див. рисунок 2.2), першим етапом є завантаження та відтворення музичних файлів. Це було реалізовано через інтерактивний інтерфейс із можливістю вибору файлу через стандартний діалоговий елемент завантаження або шляхом перетягування файлу в потрібну область інтерфейсу. Для інтуїтивного інтерфейсу завантаження використовувалась JavaFX, а для аналізу та відтворення аудіо – Java Sound API. Якщо завантажена композиція не відповідала дозволеним форматам (наприклад, MP3, WAV або MIDI), система показувала відповідне повідомлення про помилку. У разі відповідного формату, програма виконувала попередню обробку файлу, яка включала отримання його основних параметрів, таких як формат файлу, час, якість та назва композиції.

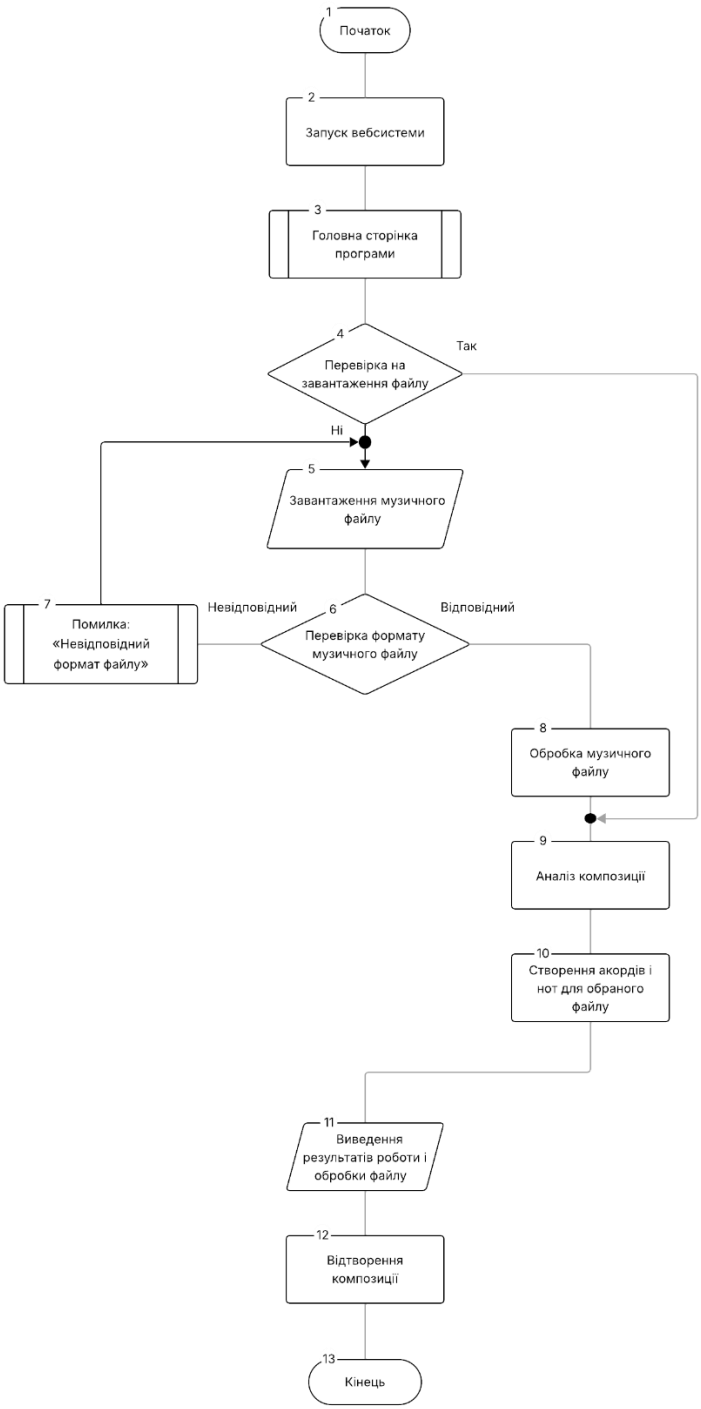

Рисунок 2.2 – Алгоритм роботи системи

Наступним кроком у процесі оцінки якості зображень стало визначення методу аналізу. Програмний модуль, який відповідає за розбиття музики на складові (ноти, акорди).

Для зберігання та обробки даних використовувалась локальна база IndexedDB, що дозволяла кешувати результати обчислень та уникнути повторного аналізу при повторному завантаженні. Крім того, було застосовано JFrame для управління станом додатку, що дало змогу оптимізувати роботу з великими обсягами даних.

Останнім етапом стало тестування програмних модулів, яке включало відтворення композиції у програмі та програвання нот та акордів.

Використання JavaFX дало змогу забезпечити швидку взаємодію користувача з програмним інтерфейсом, що дозволяло отримувати результати аналізу в режимі реального часу. Також застосування цієї бібліотеки сприяло покращенню продуктивності програми завдяки ефективному оновленню компонентів без потреби в перерендерингу всієї сторінки. Завдяки цьому програма працювала стабільно, швидко обробляла великі обсяги даних.

2.4 Висновки

У другому розділі було висвітлено етап розробки програмного забезпечення, призначеного для навчання гри на музичних інструментах. Детально описано створення користувацького інтерфейсу з використанням JavaFX, а також інтеграцію бібліотек для відтворення аудіо-файлів. Зокрема, було реалізовано модулі для завантаження та обробки музичних файлів.

Створено UML діаграму діяльності, що наочно демонструє послідовність взаємодії користувача з програмним продуктом. Також представлено алгоритм функціонування системи, який охоплює: завантаження, перевірку формату файлу, аналіз файлу та відтворення композиції разом із нотами та акордами.

Розроблені алгоритми гарантують якісне та ефективне навчання гри на музичних інструментах, передбачаючи опцію експорту отриманих результатів для подальшого їх використання.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ДОДАТКУ ДЛЯ НАВЧАННЯ ГРИ НА МУЗИЧНИХ ІНСТРУМЕНТАХ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення TuMusic

Одним з ключових чинників при виборі технологічного стеку була потреба у створенні інтерактивного вебінтерфейсу, що дозволяв би завантажувати та відтворювати музичні композиції, прослуховувати та проводити навчання. З огляду на ці вимоги, було вирішено використовувати JavaFX [14]. Дана бібліотека пропонує високу швидкість роботи завдяки віртуальному DOM, компонентний підхід до архітектури, а також гарно інтегрується з різноманітними бібліотеками та сервісами для обробки даних.

Для роботи зі звуком використано бібліотеку Java Sound API (пакет javax.sound), яка забезпечила запис, відтворення аудіо та обробку MIDI-сигналів. Бібліотека JFugue допомогла генерувати мелодії програмно та інтегрувати їх у навчальні вправи. Для аналізу висоти тону та ритму застосовано TarsosDSP, що дозволило створити тюнер для гітари.

Інтерфейс програми побудовано на JavaFX, що надало сучасні інструменти для створення інтерактивних елементів: кнопок, слайдерів гучності, візуалізації нотних партій. Використано FXML для розмітки та CSS для стилізації, що спростило розробку UI.

Для роботи з аудіо було розглянуто Java Sound API, TarsosDSP та PureData. Java Sound API була обрана як базова бібліотека через її інтеграцію з платформою Java, стабільність та достатні можливості для базової обробки звуку. TarsosDSP була додана для розширених функцій, таких як аналіз висоти тону та ритму, через її спеціалізацію на музичних додатках. PureData була відхилена через складність інтеграції з Java-додатком.

Для відображення нотних записів і гітарних табулатур застосовано бібліотеку ABC4J, яка конвертує текстові ноти (формат ABC) у графічні елементи. Додатково використано JMusic для роботи з MIDI-партитурами.

Остаточний вибір інструментів був зроблений на основі балансу між продуктивністю, простотою інтеграції, доступністю документації та можливістю подальшого масштабування. Java у поєднанні з JavaFX та TarsosDSP забезпечила оптимальне рішення для створення кросплатформенного, інтерактивного та ефективного навчального додатку для гри на гітарі. Використання сучасних бібліотек дозволило зосередитись на логіці додатку, а не на низькорівневій реалізації аудіо чи графіки, що прискорило процес розробки та поліпшило якість кінцевого продукту.

Модульні тести написані за допомогою JUnit 5. Для створення моків (наприклад, імітації аудіо-драйвера) використано Mockito. Інтеграційні тести з TestFX перевірили коректність UI, а JaCoCo забезпечив аналіз покриття коду.

3.2 Розробка модуля визначення характеристик завантаженої композиції

Модуль аналізує завантажену користувачем композицію для автоматичного визначення її ключових характеристик, що необхідні для ефективного навчання. Він використовує бібліотеку TarsosDSP [15] для обчислення основного тону (частоти) кожної ноти, що дозволяє ідентифікувати акорди та мелодійні лінії. Для розпізнавання ритму та темпу застосовується алгоритм Beat Detection, який аналізує часові інтервали між звуковими імпульсами (див. рисунок 3.1).

```
private void checkVersion(){
        for(int i = 0;i < this.arguments.length;i++){
                for(int j = 0;j < OPTION_VERSION.length;j++){
                        if(this.arguments[i].equals(OPTION_VERSION[j])){
                                print(TGApplication.NAME + " - " + TGVersion.CURRENT.getVersion());

                                this.processAndExit = true;
                        }
                }
        }
}

private void checkSystemInfo(){
        for(int i = 0;i < this.arguments.length;i++){
                for(int j = 0;j < OPTION_JRE_INFO.length;j++){
                        if(this.arguments[i].equals(OPTION_JRE_INFO[j])){
                                print("System Info:");
                                print("-> OS-Name:           " + System.getProperty("os.name"));
                                print("-> OS-Arch:           " + System.getProperty("os.arch"));
                                print("-> OS-Version:        " + System.getProperty("os.version"));
                                print("-> JVM-Name:          " + System.getProperty("java.vm.name"));
                                print("-> JVM-Version:       " + System.getProperty("java.vm.version"));
                                print("-> JVM-Vendor:        " + System.getProperty("java.vm.vendor"));
                                print("-> Java-Version:      " + System.getProperty("java.version"));
                                print("-> Java-Vendor:       " + System.getProperty("java.vendor"));
                                print("-> Java-Home:         " + System.getProperty("java.home"));
                                print("-> Java-Class-Path:   " + System.getProperty("java.class.path"));
                                print("-> Java-Library-Path: " + System.getProperty("java.library.path"));

                                this.processAndExit = true;
                        }
```

Рисунок 3.1 – Функція відтворення головного меню

Функція амплітудного аналізу (через Java Sound API) визначає динамічні відтінки (гучність, атаку, затухання звуку), що важливо для відтворення експресії. Додатково модуль перевіряє наявність шумів або спотворень за допомогою FFT-перетворення, щоб запропонувати користувачу оптимізувати якість аудіо.

Модуль інтегрований з основним інтерфейсом через JavaFX, що забезпечує візуалізацію результатів у вигляді спектрограм або нотного листа. Для зберігання аналізованих даних використовується SQLite [16], що дозволяє порівнювати поточні результати з попередніми сесіями.

Розробка модуля скоротила час на підготовку матеріалів для навчання та персоналізувала процес гри, адаптуючи його під можливості кожного музиканта (див. рисунок 3.2).

```
public void updateCache(final boolean updateItems, final TGAbstractContext sourceContext){
        this.lock();
        this.getEditorCache().updateEditMode();
        this.unlock();

        if( updateItems ){
                getEditorManager().updateSelection(sourceContext);
        }
        getEditorManager().redraw(sourceContext);
}

public void redrawPlayingMode(){
        if(!isDisposed() && !this.isLocked()){
                this.lock();
                this.getEditorCache().updatePlayMode();

                if( this.getEditorCache().shouldRedraw() ) {
                        this.getEditorManager().redrawPlayingNewBeat();
                } else {
                        this.getEditorManager().redrawPlayingThread();
                }

                this.unlock();
        }
}
```

Рисунок 3.2 – Функція redrawPlayingMode, яка застосовує алгоритм Beat Detection

Таким чином, модуль «redrawPlayingMode» забезпечує безпечну та швидку обробку файлів, і подальше відтворення композицій та нот і акордів.

3.3 Розробка модуля точного налаштуввання музичних параметрів

Модуль точного налаштування музичних параметрів був створений для забезпечення детального контролю над аудіохарактеристиками композиції під час навчання. Він дозволяє користувачам точно регулювати тембр, гучність, темпи та інші ключові параметри, що робить процес навчання більш ефективним та персоналізованим (див. рисунок 3.3).

Для реалізації модуля була використана бібліотека Java Sound API, яка забезпечила низькорівневий доступ до аудіопотоків, дозволивши точно керувати рівнями гучності, балансом каналів та частотами. Бібліотека TarsosDSP була інтегрована для аналізу та корекції висоти звуку, що особливо важливо для налаштування інструмента під конкретні ноти або акорди.

```
public boolean isInterceptable(String id, String[] actionIds) {
        for(String actionId : actionIds) {
                if( actionId.equals(id) ) {
                        return true;
                }
        }
        return false;
}

public boolean intercept(final String id, final TGActionContext context) throws TGActionException {
        if(!this.isDone(context)) {
                if( this.isInterceptable(id, INTERCEPTABLE_READ_ACTIONS)) {
                        return this.processSettingsHandler(id, context, TGPersistenceSettingsMode.READ);
                }
                if( this.isInterceptable(id, INTERCEPTABLE_WRITE_ACTIONS)) {
                        return this.processSettingsHandler(id, context, TGPersistenceSettingsMode.WRITE);
                }
        }
        return false;
}
```

Рисунок 3.3 – Функція перевірки формату файла

Функція тембрової корекції дозволяє користувачам змінювати характер звучання гітари, імітуючи різні типи інструментів за допомогою бібліотеки JSyn. Це дає можливість експериментувати з різними звуковими палітрами під час навчання.

Модуль також включає прецизійний метроном, розроблений з використанням алгоритмів точного таймінгу, що дозволяє встановлювати темп з кроком у 1 BPM. Це допомагає користувачам поступово збільшувати швидкість гри, покращуючи техніку.

```
public URI getCurrentURI(){
        TGDocument document = TGDocumentListManager.getInstance(this.context).findCurrentDocument();
        if( document != null ) {
                return document.getUri();
        }
        return null;
}

public String getFilePath(URI uri){
        if( TGFileUtils.isLocalFile(uri) ){
                return new File(uri).getParent();
        }
        return null;
}

public boolean isNewFile(){
        return (this.getCurrentURI() == null);
}

public boolean isLocalFile(){
        URI uri = getCurrentURI();

        return (uri != null && TGFileUtils.isLocalFile(uri));
}

private String decode(String url){
        try {
                return URLDecoder.decode(url, "UTF-8");
        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        }
        return url;
}
```

Рисунок 3.4 – Отримання даних про аудіофайл

Для візуалізації параметрів у реальному часі був застосований JavaFX, який забезпечив інтерактивні графіки та еквалайзери. Користувачі можуть бачити, як зміни параметрів впливають на звучання, що робить процес навчання більш наочним (див. рисунок 3.5).

```
public boolean processSettingsHandler(final String id, final TGActionContext actionContext) {
        actionContext.setAttribute(ATTRIBUTE_DONE, Boolean.TRUE);

        TGSongStreamProvider provider = actionContext.getAttribute(ATTRIBUTE_STREAM_PROVIDER);
        if( provider != null ) {
                TGSongStreamSettingsHandler handler = TGSongStreamAdapterManager.getInstance(this.context).findSet
                if( handler != null ) {
                        TGSongStreamContext streamContext = actionContext.getAttribute(ATTRIBUTE_STREAM_CONTEXT);
                        if( streamContext == null ) {
                                streamContext = new TGSongStreamContext();
                                streamContext.addContext(actionContext);
                                actionContext.setAttribute(ATTRIBUTE_STREAM_CONTEXT, streamContext);
                        }

                        handler.handleSettings(streamContext, createExecuteActionThread(id, actionContext));

                        return true;
                }
        }
        return false;
}

public Runnable createExecuteActionThread(final String id, final TGActionContext context) {
        return new Runnable() {
                public void run() {
                        new Thread(createExecuteActionRunnable(id, context)).start();
                }
        };
}
```

Рисунок 3.5 – Перевірка доданих аудіофайлів

Модуль інтегрований з системою збереження профілів, що дозволяє зберігати індивідуальні налаштування для різних композицій або вправ. Це забезпечує швидкий доступ до оптимальних параметрів під час кожного сеансу гри.

Основна мета модуля – надати музикантам інструменти для тонкого налаштування звуку, що дозволяє адаптувати навчальний процес до індивідуальних потреб. Наприклад, користувач може точно підігнати тембр під свій стиль гри або виставити ідеальний темп для відпрацювання складної партії. Це значно підвищує ефективність тренувань і робить їх більш цілеспрямованими, що зображено на рисунку 3.6.

```
public void createPopupMenu() {
        UIWindow window = TGWindow.getInstance(this.context).getWindow();
        if(!window.isDisposed()) {
                if( this.popupMenu == null || this.popupMenu.isDisposed() ){
                        this.popupMenu = TGApplication.getInstance(this.context).getFactory().createPopupMenu(window);
                }
                List<UIMenuItem> items = this.popupMenu.getItems();
                for(UIMenuItem uiMenuItem : items){
                        uiMenuItem.dispose();
                }

                this.loadedPopupMenuItems.clear();
                this.loadedPopupMenuItems.add(new EditMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new CompositionMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new TrackMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new MeasureMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new BeatMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new MarkerMenuItem(this.popupMenu));
                this.loadedPopupMenuItems.add(new TransportMenuItem(this.popupMenu));
                this.showMenuItems(this.loadedPopupMenuItems);
        }
}

private void showMenuItems(List<TGMenuItem> items){
        Iterator<TGMenuItem> it = items.iterator();
        while(it.hasNext()){
                TGMenuItem item = (TGMenuItem)it.next();
                item.showItems();
        }
}
```

Рисунок 3.6 – Головне меню програми

Даний модуль був реалізований за допомогою бібліотеки JSyn [17], що дало можливість програвати різні мелодії, звуки (див. рисунок 3.7).

```
public static String getProperty(String key,String[] arguments) {
        return  TuxGuitar.getInstance().getLanguageManager().getProperty(key,arguments);
}

public boolean isDisposed(){
        return (TGApplication.getInstance(this.context).isDisposed() || TGWindow.getInstance(this.context).isDisposed());
}

public void updateSong(){
        this.lock();
        this.getEditorCache().reset();
        this.getEditorManager().updateSong();
        this.unlock();
}

public void playBeat( final TGBeat beat ){
        TGThreadManager.getInstance(this.context).start(new Runnable() {
                public void run() throws TGException {
                        if(!isDisposed() && !getPlayer().isRunning() ){
                                getPlayer().playBeat(beat);
                        }
                }
        });
}
```

Рисунок 3.7 – Функція «playBeat», яка реалізована за допомогою бібліотеки JSyn

Даний модуль став ключовим інструментом для персоналізації процесу навчання гри на музичних інструментах. Він надає музикантам точний контроль над усіма аспектами звучання, від частоти нот до тембру й ритмічного малюнку. Інтеграція таких бібліотек, як Java Sound API, TarsosDSP та JSyn, дозволила створити потужний інструментарій для аналізу та корекції музичних параметрів у реальному часі.

У результаті модуль не лише спростив процес навчання, але й зробив його більш ефективним, дозволяючи музикантам зосередитись на вдосконаленні своєї майстерності, а не на технічних аспектах звукорегулювання. Це істотно розширює можливості програми як професійного навчального інструменту для гітаристів будь-якого рівня підготовки.

3.4 Розробка модуля відтворення нот та акордів

Модуль відтворення нот та акордів був створений як ключовий інструмент для навчання гри на гітарі, забезпечуючи точне аудіовідтворення музичних елементів. Основна мета модуля – надати користувачам зразкове звучання нот, акордів та їх послідовностей, що є критично важливим для розвитку музичного слуху, вивчення аплікатур та вдосконалення техніки гри.

Для реалізації якісного звукового відтворення була використана бібліотека Java Sound API, яка забезпечила низькорівневий доступ до аудіосистеми комп'ютера. Це дозволило реалізувати точне керування тривалістю, гучністю та тембром кожного звуку. Для роботи з MIDI-даними була інтегрована бібліотека JFugue, що спростила генерацію та відтворення нотних послідовностей у стандартній музичній нотації (див. рисунок 3.6).

```
private static String getUserSharedPath(){
        // Look for the system property
        String userSharePath = System.getProperty(TG_USER_SHARE_PATH);

        // Use configuration path as default.
        if( userSharePath == null){
                userSharePath = (getDefaultUserAppDir() + File.separator + "cache");
        }

        // Check if the path exists
        File file = new File(userSharePath);
        if(!isExistentAndReadable( file )){
                tryCreateDirectory( file );
        }
        return userSharePath;
}

private static String[] getStaticSharedPaths(){
        String staticSharedPaths = new String(PATH_USER_SHARE_PATH);
        String staticSharedPathsProperty = System.getProperty(TG_SHARE_PATH);
        if( staticSharedPathsProperty != null ){
                staticSharedPaths += (File.pathSeparator + staticSharedPathsProperty);
        }
        return staticSharedPaths.split(File.pathSeparator);
}
```

Рисунок 3.6 – Функція «getUserSharedPath», викликана за допомогою бібліотеки TarsosDSP

Модуль включає функцію пошагового відтворення, яка дозволяє слухати окремі ноти акорду або мелодії з заданим інтервалом. Це особливо корисне для початківців, які тільки вивчають аплікатуру. Для більш досвідчених музикантів реалізовано режим циклічного відтворення, що дає можливість багаторазово прослуховувати складні пасажі для точного запам'ятовування.

Важливою частиною модуля є візуальне супроводження звуку, реалізоване за допомогою JavaFX. Користувач бачить нотний запис або табулатуру, яка підсвічується синхронно з відтворенням, що значно покращує сприйняття музичного матеріалу. Для гітарних акордів додатково відображається схема постановки пальців на грифі.

```
public void updateTabItems() {
	if(!this.isDisposed()) {
		this.ignoreEvents = true;
		this.disposeOrphanItems();

		TGDocumentListManager documentManager = TGDocumentListManager.getInstance(this.context);
		TGDocument currentDocument = documentManager.findCurrentDocument();
		List<TGDocument> documents = documentManager.getDocuments();
		for(int i = 0 ; i < documents.size() ; i ++) {
			TGDocument document = documents.get(i);

			UITabItem uiTabItem = this.findTabItem(i);
			uiTabItem.setText(this.createTabItemLabel(document));
			uiTabItem.setData(TGDocument.class.getName(), document);
			if( currentDocument != null && currentDocument.equals(document) ) {
				this.tabFolder.setSelectedIndex(i);
			}
		}

		this.currentUnsaved = currentDocument.isUnsaved();
		this.ignoreEvents = false;
	}
}

public void disposeOrphanItems() {
	if(!this.isDisposed()) {
		List<TGDocument> documents = TGDocumentListManager.getInstance(this.context).getDocuments();
		while( this.tabItems.size() > documents.size() ) {
			int index = (this.tabItems.size() - 1);
			UITabItem uiTabItem = this.tabItems.get(index);
			uiTabItem.dispose();
			this.tabItems.remove(index);
		}
	}
```

Рисунок 3.7 – Налаштування музичних інструментів

Для емуляції різних тембрів гітари (нейлонові, сталеві струни, електрогітара) використана бібліотека JSyn, яка дозволила програмно змінювати характеристики звуку. Це дає можливість почути, як одна й та сама нота чи акорд звучить на різних інструментах (див. рисунок 3.8).

```
public void gotoPlayerPosition(){
        MidiPlayer player = TuxGuitar.getInstance().getPlayer();
        TGMeasureHeader header = getSongManager().getMeasureHeaderAt(getSong(), MidiTickUtil.getStart
        if(header != null){
                player.setTickPosition(MidiTickUtil.getTick(header.getStart()));
        }
        TuxGuitar.getInstance().getTablatureEditor().getTablature().getCaret().goToTickPosition();

        TuxGuitar.getInstance().updateCache(true);
}

public void play(){
        MidiPlayer player = TuxGuitar.getInstance().getPlayer();
        if(!player.isRunning()){
                try{
                        player.getMode().reset();
                        player.play();
                }catch(MidiPlayerException e){
                        TGErrorManager.getInstance(this.context).handleError(e);
                }
        }else{
                player.pause();
        }
}

public void stop(){
        MidiPlayer player = TuxGuitar.getInstance().getPlayer();
        if(!player.isRunning()){
                player.reset();
                this.gotoPlayerPosition();
        }else{
                player.reset();
        }
```

Рисунок 3.8 – Функція «play», яка відтвоює звуки та мелодії

Модуль інтегрований з системою навчальних вправ, автоматично підбираючи оптимальну швидкість відтворення залежно від рівня складності матеріалу та прогресу користувача. При цьому враховується індивідуальний темп навчання, що дозволяє кожному музиканту рухатися з комфортною для нього швидкістю.

Додатково реалізовано функцію порівняльного аналізу, коли система може одночасно відтворювати еталонний звук і запис гри користувача, що сприяє розвитку слуху та точності виконання. Для цього було використано алгоритми аудіопорівняння з бібліотеки TarsosDSP.

Модуль відтворення нот та акордів став невід'ємною частиною навчального процесу, поєднуючи в собі функції тренера, метронома та аудіо-прикладу. Він

дозволяє не тільки чути правильне звучання, але й бачити його графічне представлення, що значно прискорює процес освоєння інструменту та покращує якість навчання.

3.5 Розробка інтерфейсу програмного застосунку

При розробці користувацького інтерфейсу програмного забезпечення особливу увагу було приділено створенню інтуїтивно зрозумілого, візуально привабливого та технологічно сучасного середовища для користувача.

Для реалізації інтерактивних елементів та динамічної поведінки інтерфейсу було використано бібліотеку JavaFX, яка забезпечує ефективне оновлення компонентів та зручну обробку подій користувача. Стилізація елементів інтерфейсу здійснювалася за допомогою фреймворку Tailwind CSS, що дозволило швидко створювати адаптивний та візуально узгоджений дизайн завдяки використанню готових класів стилів.

Початковий етап взаємодії користувача із системою починається із головного екрану програми (див. рисунок 3.9).

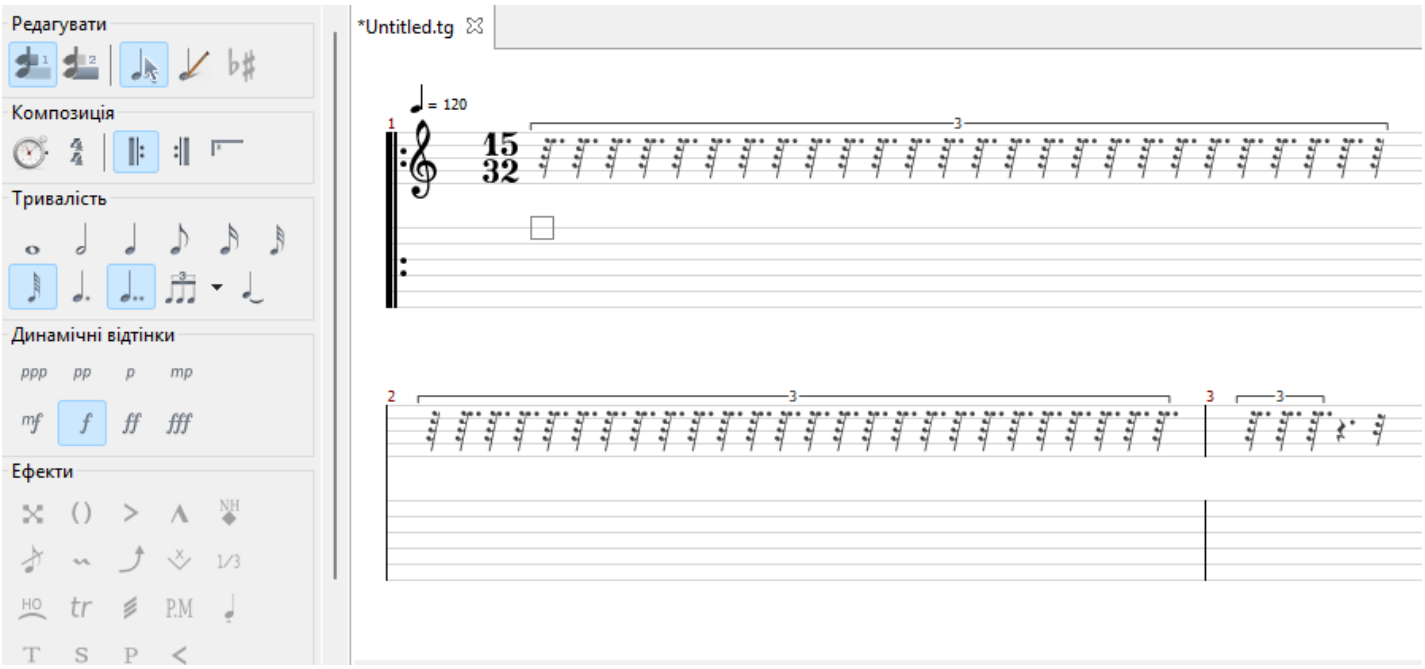

Рисунок 3.9 – Головна сторінка програми

Дизайн цієї сторінки, створеної з використанням JavaFX та Tailwind CSS, виконано у світлих тонах, що створює відчуття відкритості та простоти. Передбачено можливість скористатися функціоналом програми без попереднього додавання аудіофайлу, що забезпечує швидкий доступ до основних можливостей.

Наступним кроком є вибір музичного інструмента, у відокремленому вікні програми JavaFX та стилізовані за допомогою Tailwind CSS. На сторінці показано усі можливі налаштування для музичних інструментів (див. рисунок 3.10).

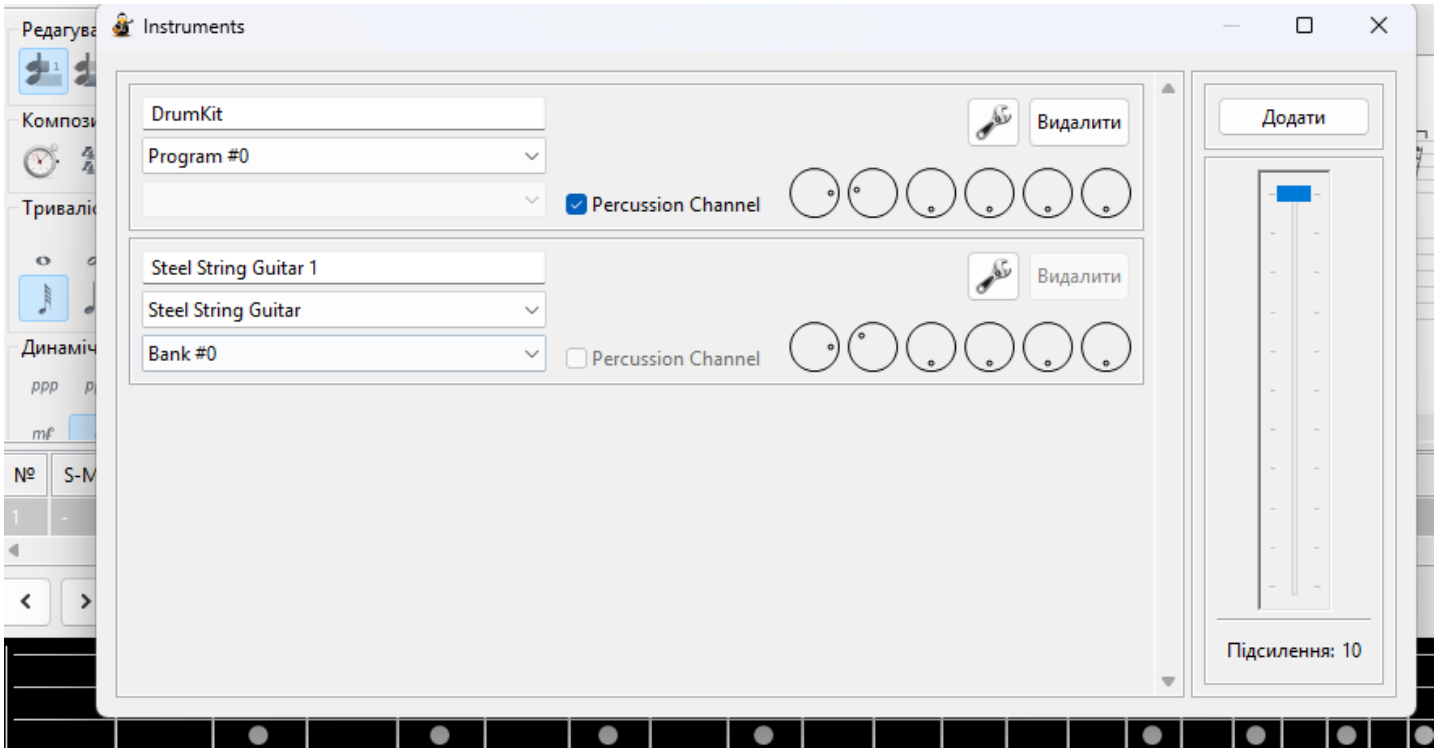

Рисунок 3.10 – Вікно програми, де можна налаштувати необхідний інструмент

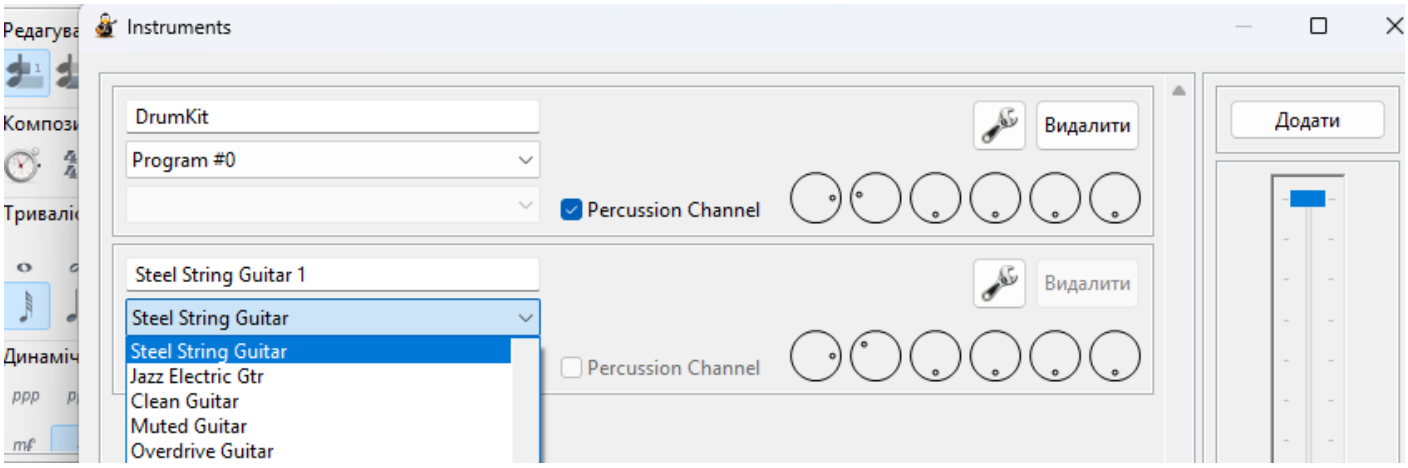

Рисунок 3.11 – Меню вибору необхідного музичного інструменту

У меню вибору композицій, можна додати, видалити і обрати потрібний користувачеві аудіофайл (див. рисунок 3.12). Після видалення, композиція зникне зі списку, а при додаванні користувачеві потрібно обрати місцезнаходження композиції у системі.

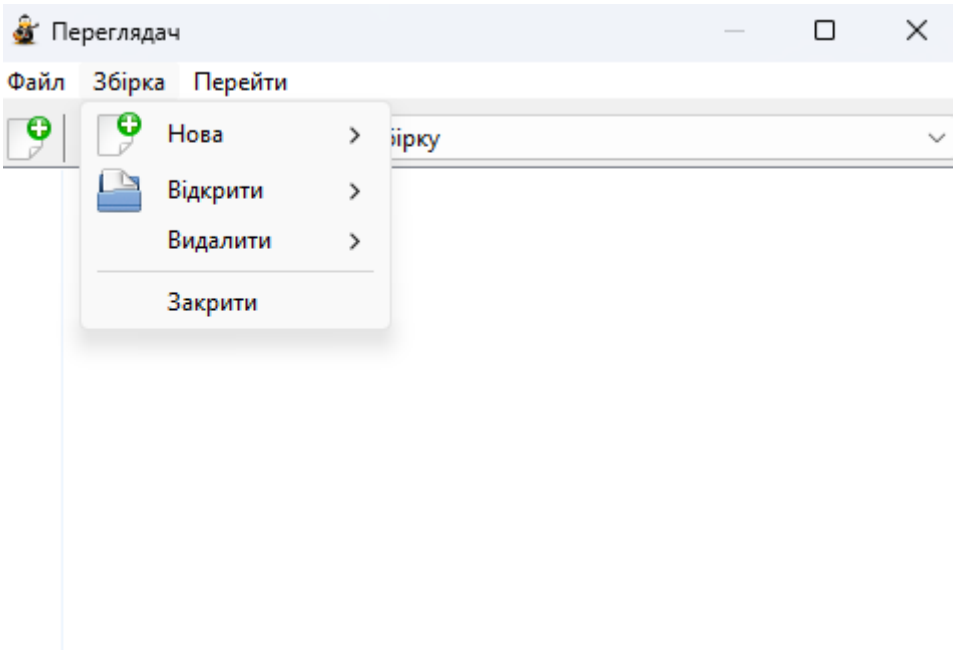

Рисунок 3.12 – Меню доданих аудіофайлів

Також була розроблена додаткова графа, яка вказує на правильне та вчасне попадання в ритм композиції. Червоний квадрат означає, що користувач нажав вчасно, чорний – це промах, а сірі, тобто порожні квадрати, означають, що нічого не було зроблено (див. рисунок 3.13)

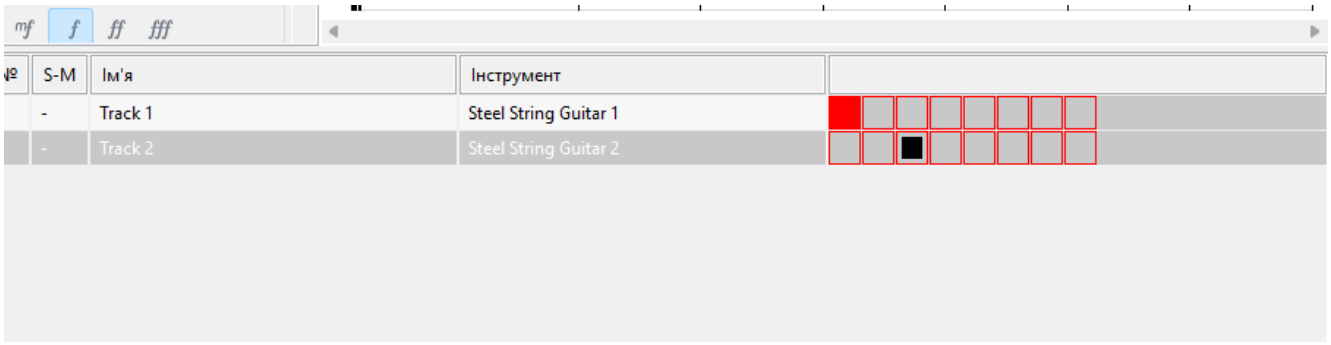

Рисунок 3.13 – Додаткова графа

Таким чином, розроблений користувацький інтерфейс, завдяки використанню сучасних вебтехнологій, таких як JavaFX для інтерактивності та керування станом, JSyn для відображення акордів та нот, прагне бути інтуїтивно зрозумілим, візуально привабливим, технологічно сучасним та максимально зручним для виконання основних завдань програмного забезпечення.

3.6 Висновки

У третьому розділі описано повний цикл розробки програмних модулів, призначених для системи навчання гри на музичних інструментах, що отримала назву TuMusic. У відповідних підрозділах розглядаються теоретичні підґрунтя та практичні аспекти впровадження функціоналу.

У сукупності, всі підрозділи розділу 3 спрямовані на створення повноцінної, зручної та наочної системи для навчання гри на музичних інструментах. Розроблені модулі забезпечують функціональність як базового рівня, так і експертну оцінку якості. Це робить систему придатною для використання як у навчальних, так і у підвищенні власних навичок.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Системне тестування – це тестування програмного забезпечення, що виконується на повній, інтегрованій системі, з метою перевірки відповідності системи вихідним вимогам, як функціональним, так і не функціональним. Метою системного тестування є виявлення розбіжностей між готовою системою та початковими цілями її розробки. У процесі беруть участь сама розроблена програмна система, кінцеві цілі її створення та вся супутня документація. На відміну від функціонального тестування, яке спирається на зовнішні специфікації, вони не мають значення під час системного тестування [18].

Стосовно програмної системи, призначеної для кількісного оцінювання якості візуальних даних, системне тестування охоплює увесь технологічний ланцюжок обробки інформації. Воно включає етапи запуску та завантаження графічних файлів різних форматів та розмірів, їх подальшу обробку згідно з визначеними алгоритмами, процедури порівняльного аналізу, кількісне визначення відповідних показників якості та візуалізацію отриманих результатів. Ключовим є підтвердження правильності використаних алгоритмів оцінки, точності проведених обчислень, адекватності представлення вихідних даних, а також стійкості системи до виникнення потенційних помилок, таких як некоректні вхідні дані або обмежені обчислювальні ресурси.

Системне тестування також передбачає імітацію реальних умов експлуатації програмного забезпечення. Так, воно охоплює перевірку продуктивності системи під час пікових навантажень, оцінку стабільності при тривалому безперервному використанні, аналіз ергономіки інтерфейсу (якщо він передбачений) та аудит захисту оброблюваних даних.

Для комплексної оцінки якості програмного забезпечення, призначеного для навчання грі на музичних інструментах, застосовується цілий спектр тестувальних методів. Серед них – функціональне тестування (перевірка відповідності

задекларованим можливостям), нефункціональне тестування (аналіз швидкодії, надійності, стресостійкості, безпеки та зручності) та регресійне тестування (контроль того, щоб нові зміни не порушували працездатність існуючих функцій).

Успішне проходження системного тестування є критично важливим етапом, оскільки саме воно забезпечує випуск стабільного та ефективного програмного рішення, здатного повністю відповідати очікуванням кінцевих користувачів.

Під час роботи програми для навчання грі на музичних інструментах система може перебувати в наступних станах:

1. Початковий стан – головний екран, на якому користувач бачить весь функціонал програмного застосунку (див. рисунок 4.1).

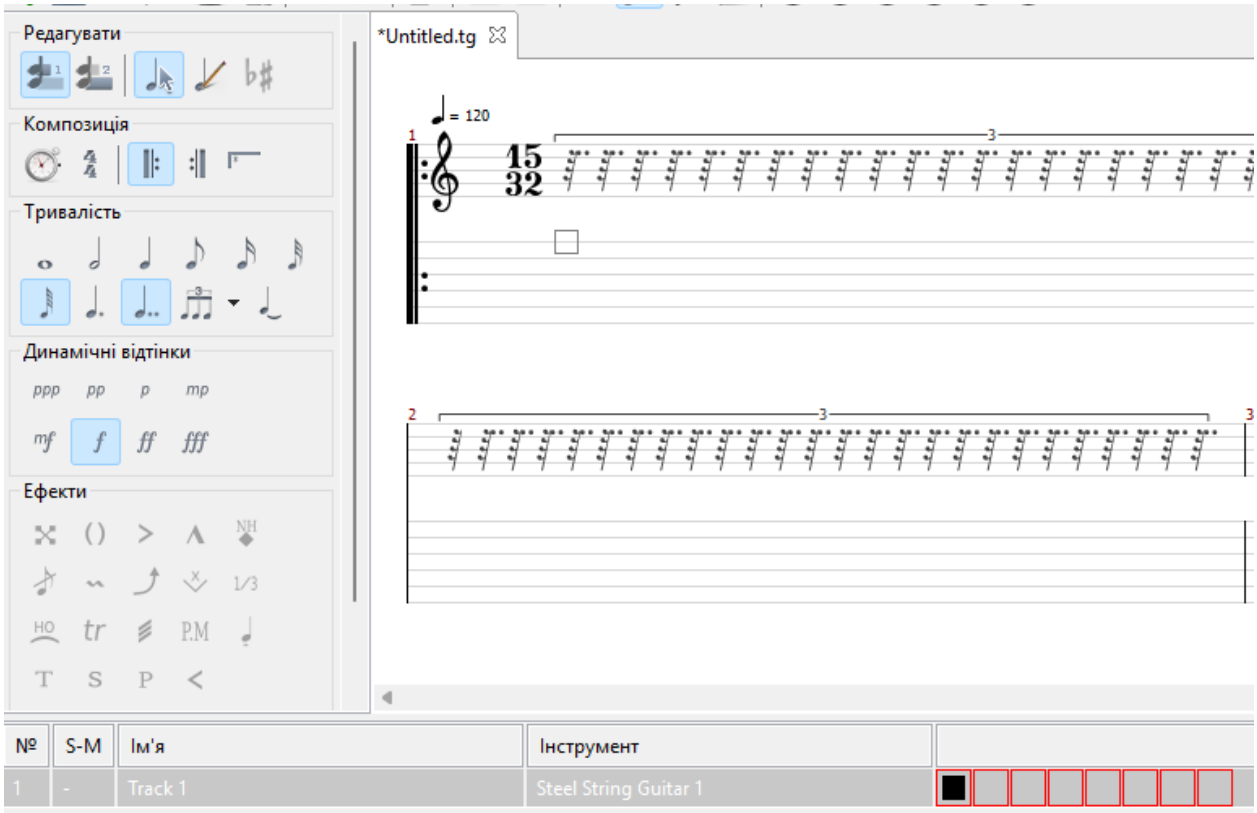

Рисунок 4.1 – Початковий стан системи

2. Завантаження аудіофайлу – користувач завантажує файл з музикою. Користувач ініціює процес завантаження музичного файлу через інтерактивний інтерфейс програми. Система надає стандартний діалог вибору файлів,

інтегрований через JavaFX, який дозволяє переглядати локальні директорії та вибирати аудіофайли у підтримуваних форматах (WAV, MP3, MIDI). Під час вибору файлу програма автоматично аналізує його розширення та структуру, щоб переконатися у коректності формату (див. рисунок 4.2).

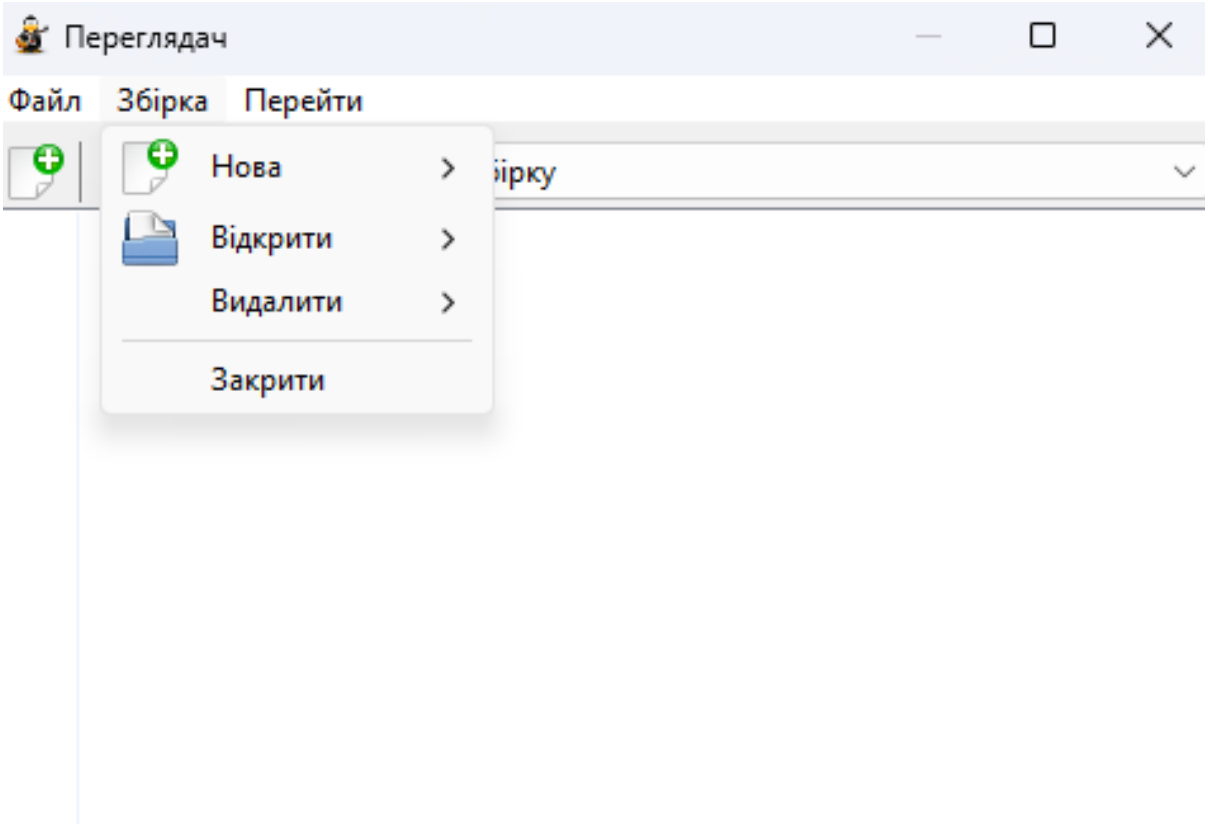

Рисунок 4.2 – Завантаження аудіофайлу

Після вибору файлу система ініціалізує процес завантаження, відображаючи індикатор прогресу, що дозволяє користувачу візуально контролювати стан операції. Для обробки аудіоданих використовується спеціалізований модуль імпорту, який заснований на Java Sound API. Цей модуль виконує низькорівневе читання аудіопотоку, перевіряє цілісність даних та конвертує вхідний сигнал у внутрішній формат програми для подальшої обробки.

Паралельно з технічним завантаженням система активує аналітичний модуль, що перевіряє базові характеристики файлу: тривалість композиції, частоту дискретизації, бітрейт та наявність метаданих. Ця інформація відображається в

інтерфейсі, даючи користувачу змогу швидко оцінити параметри завантаженої композиції. У разі виявлення потенційних проблем (наприклад, надто низька якість звуку або несумісність формату) система генерує інформативне попередження з рекомендаціями.

3. Відтворення композиції – програма готова відтворювати аудіофайл. Після успішного завантаження аудіофайлу система надає користувачу інтуїтивно зрозумілий інструментарій для керування відтворенням. Основним елементом керування є аудіоплеєр, реалізований за допомогою Java Sound API, який забезпечує стабільне декодування та відтворення звукового потоку. Інтерфейс плеєра включає візуальне відображення хвилі звуку, створене через JavaFX Canvas, що дає змогу спостерігати амплітудні характеристики композиції в реальному часі.

При активації процесу відтворення система ініціалізує аудіо потік, попередньо обробляючи дані через буферизацію для запобігання перериванням звуку. Спеціальний алгоритм балансування навантаження динамічно регулює використання системних ресурсів, гарантуючи плавне відтворення навіть на пристроях з обмеженими можливостями. Користувач має доступ до стандартних функцій керування: пуск/пауза, зупинка, перемотування вперед/назад, що реалізовані через точний контроль позиції у аудіопотоку (див. рисунок 4.3).

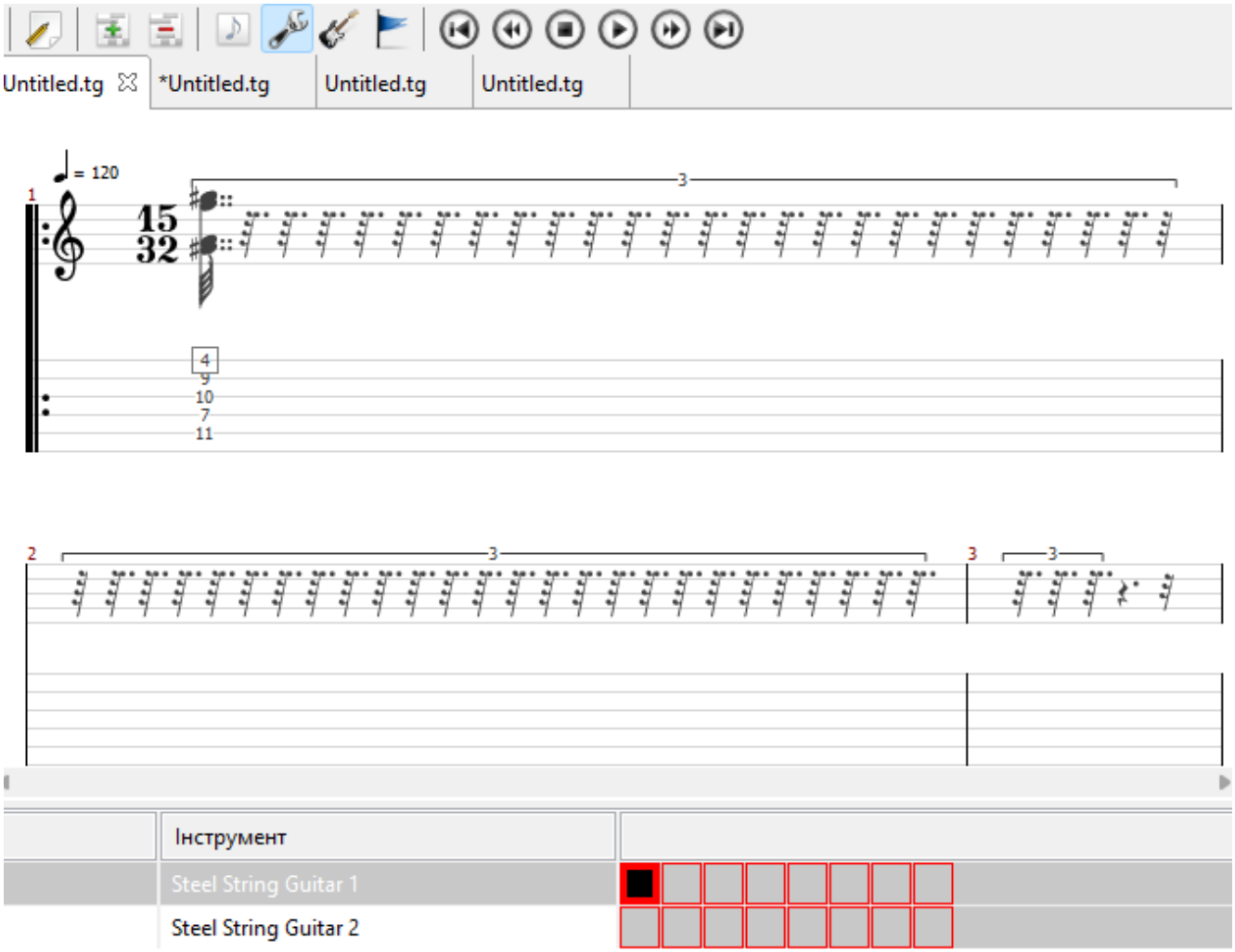

Рисунок 4.3 – Програвач для аудіофайлів

4. Налаштування музичних інструментів – користувач сам обирає, який інструмент він буде використовувати. Також можна налаштувати інструменти під свою гру.

Для акустичних гітар реалізовано інструмент точного тюнінгу, який використовує алгоритми аналізу частоти з бібліотеки TarsosDSP. Користувач може вибирати між стандартними строями (EADGBE, Drop D, Open G) або створювати власні кастомні варіанти. Система візуалізує процес налаштування через інтерактивний інтерфейс, де відображається поточна висота звуку кожної струни у реальному часі з індикацією відхилення від цільової ноти (див. рисунок 4.4).

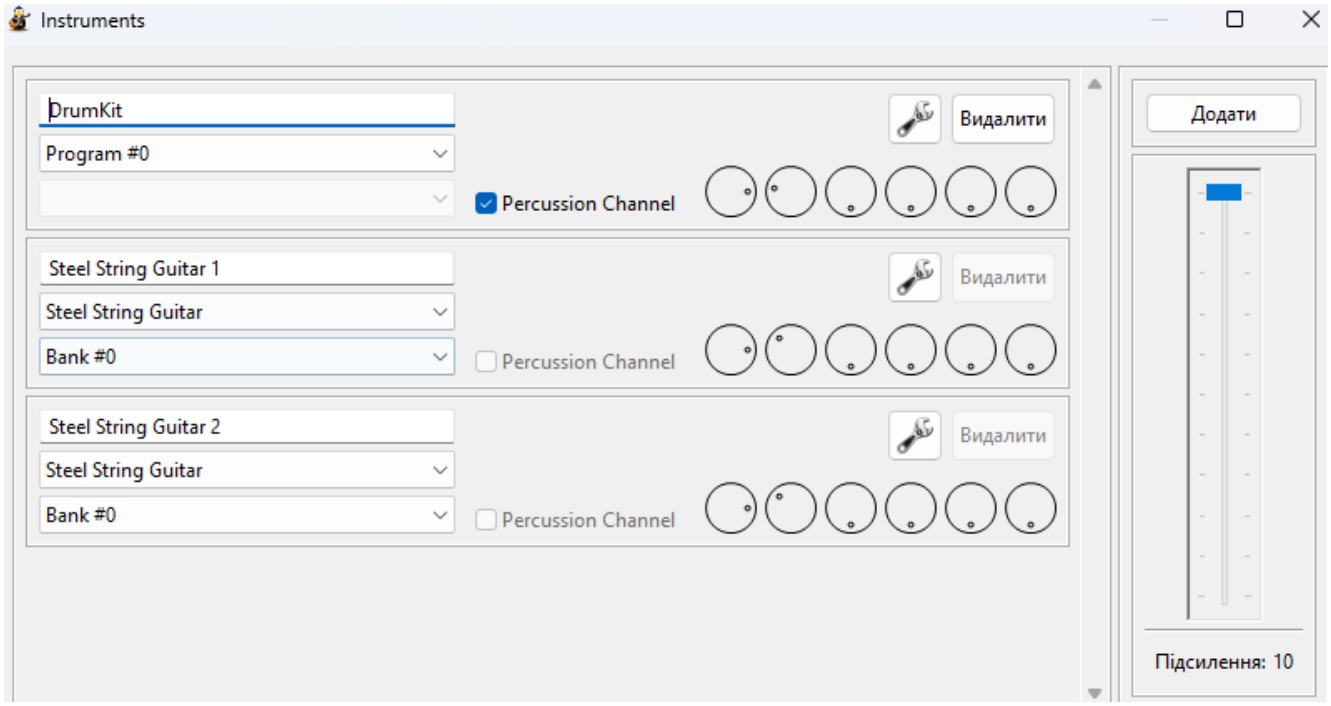

Рисунок 4.4 – Головні налаштування

Особливу увагу приділено налаштуванню тембру. Програма пропонує прецизійний 10-смуговий еквалайзер з попередньо заданими пресетами для різних музичних жанрів. Користувачі можуть змінювати параметри кожної частотної смуги, зберігати власні налаштування та ділитися ними через хмарне сховище.

Всі зміни параметрів відображаються аудіовізуально – користувач одразу чує результат налаштувань і бачить відповідні зміни на спектрограмі. Інтелектуальна система підказок аналізує вибрані параметри і пропонує оптимальні налаштування для конкретного стилю гри (див. рисунок 4.5).

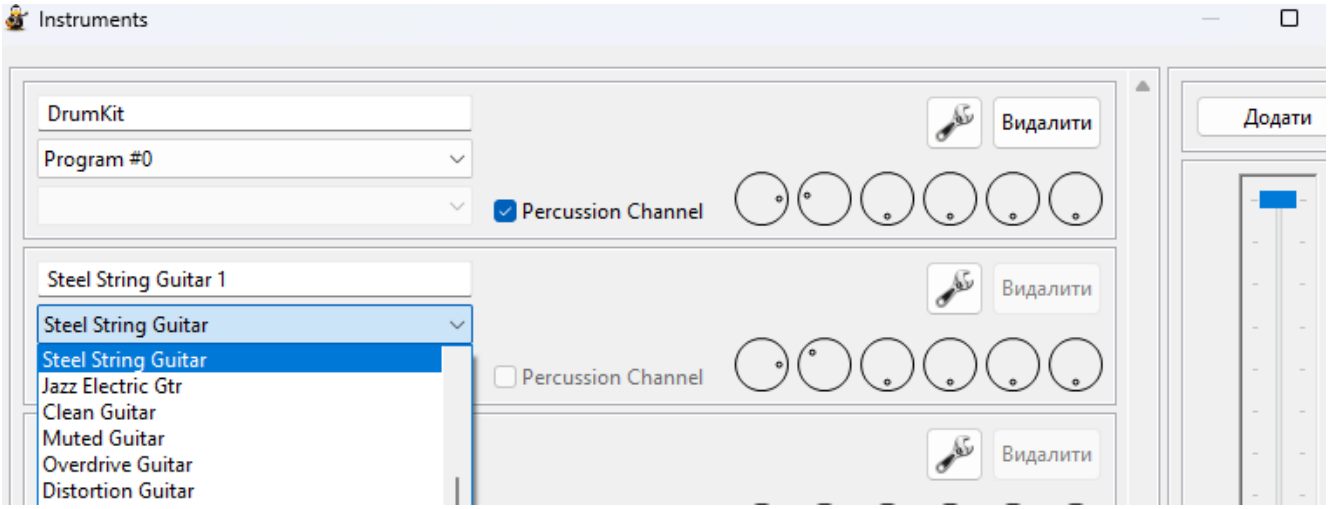

Рисунок 4.5 – Вибір музичного інструмента

Система також включає автоматичну калібровку, яка аналізує апаратні можливості пристрою користувача і оптимізує параметри обробки звуку для досягнення найкращої якості звучання при мінімальному навантаженні на систему.

5. Створення запису – користувач створює запис в якому він може записати дані про композицію. Модуль створення запису в програмі реалізує комплексний підхід до документування музичних творів, забезпечуючи структуроване зберігання всієї необхідної метаінформації. При ініціалізації процесу створення нового запису система надає спеціалізовану форму введення даних, де користувач може заповнити всі ключові атрибути композиції.

В полі "Ім'я" вказується офіційна назва музичного твору, що дозволяє ідентифікувати запис у бібліотеці. Поле "Виконавець" призначене для зазначення артиста або групи, яка виконує композицію. Для альбомних записів передбачено окреме поле "Альбом", де фіксується назва музичного альбому, до якого належить твір.

Система автоматично заповнює поле "Date" поточною датою створення запису, що дозволяє в подальшому відстежувати хронологію роботи з матеріалом. Розділ "Comments" надає можливість додати довільні текстові примітки, які можуть містити особливості виконання, історію створення композиції або іншу релевантну інформацію.

Для оцінки якості матеріалу реалізовано систему градації, де користувач може позначити статус запису ("Добре" або "Відміна"), що спрощує подальший пошук і фільтрацію матеріалів. Усі введені дані проходять перевірку на коректність формату та автоматично зберігаються у внутрішній базі даних програми.

Створений запис стає повноцінним об'єктом у системі, до якого можна прив'язувати аудіофайли, ноти, табулатури, відеоматеріали та інші пов'язані ресурси. Модуль забезпечує цілісність даних і надає інструменти для подальшого редагування метаінформації в будь-який момент (див. рисунок 4.6).

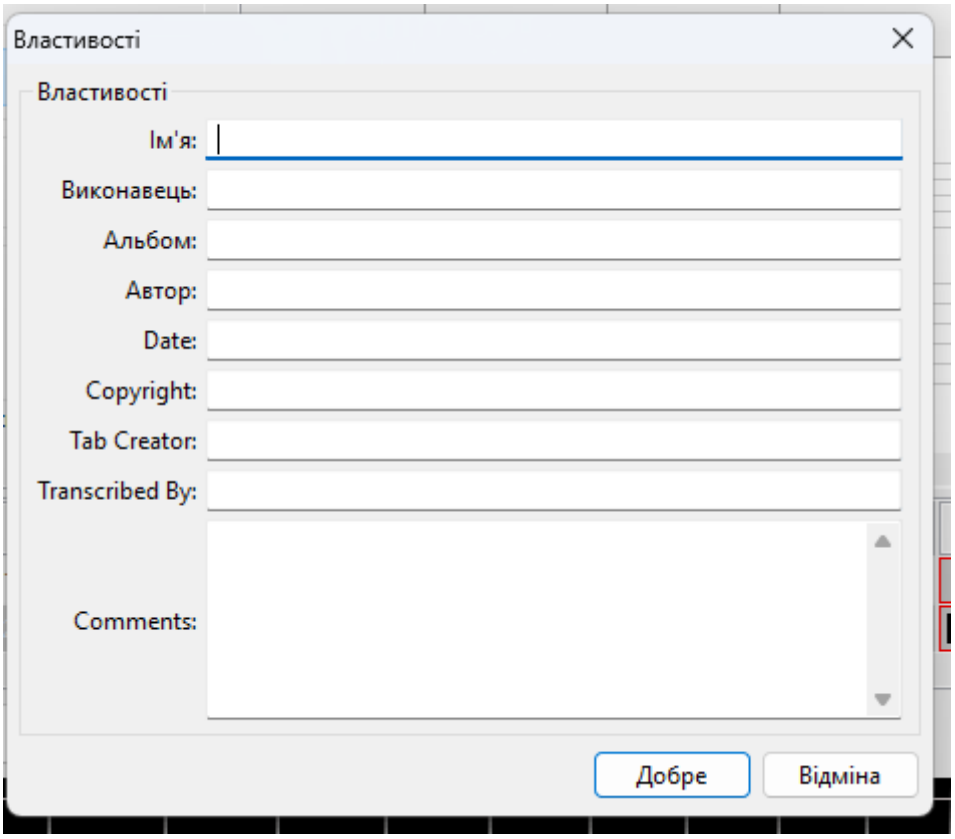

Рисунок 4.6 – Запис нотаток

6. Бокове меню для користувача – дає доступ до швидкого та зручного меню. Бокове меню в програмі для навчання грі на музичних інструментах реалізовано як багатофункціональну панель інструментів, яка забезпечує швидкий доступ до всіх основних функцій редагування та налаштувань. Розташоване збоку

інтерфейсу, воно залишається постійно доступним під час роботи з програмою, надаючи зручну навігацію без необхідності використовувати головне меню.

Меню організоване у вигляді вертикальної панелі з чітко структурованими розділами, кожен з яких відповідає за певний аспект роботи з музичною композицією. Верхній розділ "Композиція" містить інструменти для керування загальними параметрами твору, включаючи регулювання темпу і розміру. Тут же знаходиться контроль динамічних відтінків з повним спектром позначень від pianississimo (ppp) до fortississimo (fff) (див. рисунок 4.7).

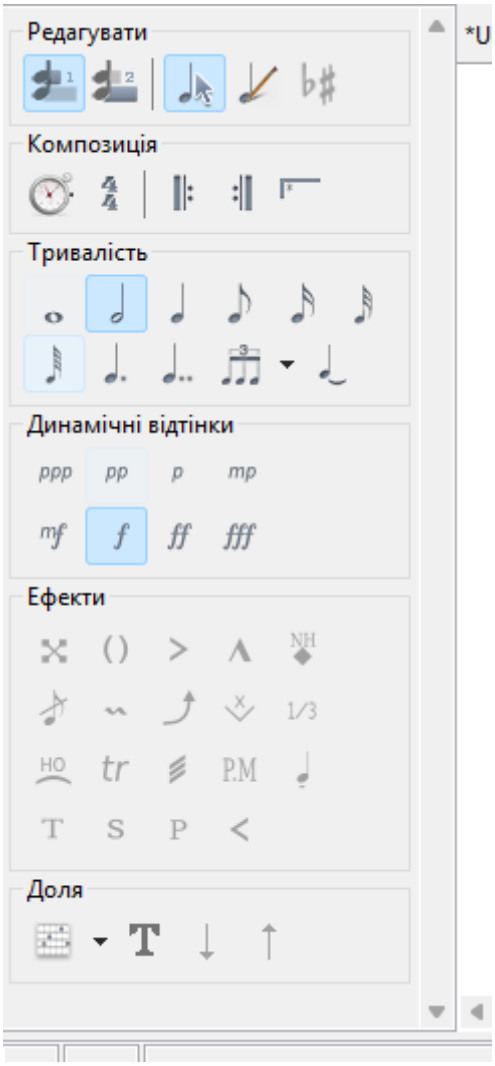

Рисунок 4.7 – Відображення результатів порівняння

7. Помилка завантаження – цей стан виникає, якщо користувач намагається завантажити файл непідтримуваного типу. Коли користувач намагається завантажити файл з непідтримуваним форматом, система автоматично виявляє цю ситуацію та переходить у спеціальний режим обробки помилки. Першим етапом є технічна перевірка розширення файлу, яка здійснюється шляхом аналізу суфіксу імені файлу та порівняння зі списком дозволених форматів (наприклад, .gp5, .midi, .wav, .mp3). Якщо розширення не відповідає жодному з підтримуваних варіантів, система блокує спробу завантаження до завершення повної діагностики.

8. Коли користувач намагається активувати функції програми, які вимагають підключеного музичного інструменту, система проводить автоматичну перевірку наявності активного аудіовходу. Ця перевірка здійснюється через Java Sound API, яке сканує доступні аудіоінтерфейси та MIDI-пристрої. Якщо жоден пристрій вводу не виявлено або підключений інструмент не відповідає вимогам програми, система генерує спеціалізоване повідомлення про помилку.

Повідомлення про помилку відображається у вигляді модального вікна з чіткою візуальною ієрархією елементів. Заголовок вікна містить іконку попередження та короткий опис проблеми: "Музичний інструмент не виявлено". Основна частина повідомлення детально пояснює причину помилки: "Для використання цієї функції необхідно підключити гітару або MIDI-контролер до аудіоінтерфейсу".

Таким чином, підрозділ навчання гри на музичних інструментах проходить через різні стани в залежності від дій користувача та результатів обробки. Система обробляє помилки, пов'язані з непідтримуваними типами файлів, та потенційні проблеми з музичними інструментами.

4.2 Розробка інструкції користувача

Дана інструкція описує процес використання підсистеми для навчання гри на музичних інструментах, включаючи технічні вимоги для її запуску та покрокову інструкцію з використання основного функціоналу.

Створення інструкції користувача є завершальним етапом розробки програмного забезпечення, який забезпечує ефективне освоєння програми користувачами різних рівнів підготовки. Інструкція розробляється як комплексний довідковий документ, що поєднує текстові пояснення, візуальні елементи та практичні приклади.

Окремий розділ присвячений навчальним можливостям програми. Він включає рекомендації щодо ефективного використання вбудованих уроків, тренувальних режимів та інструментів аналізу гри. Тут же містяться поради щодо складання індивідуального плану навчання та відстеження прогресу.

Інструкція містить розділ з відповідями на поширені запитання, де зібрані рішення типових проблем, таких як відсутність звуку, затримки при грі чи проблеми із підключенням інструменту. Кожна проблема описується з поясненням можливих причин та детальним алгоритмом їх усунення.

Кожен функціональний елемент супроводжується зрозумілим описом його призначення, варіантів використання та поширених сценаріїв застосування. Для складних функцій, таких як створення власних вправ чи робота з ефектами, наводяться розширені пояснення з прикладами.

Окремий розділ присвячений навчальним можливостям програми. Він включає рекомендації щодо ефективного використання вбудованих уроків, тренувальних режимів та інструментів аналізу гри.

Для коректної роботи підсистеми для навчання гри на музичних інструментах необхідно забезпечити відповідність апаратного та програмного забезпечення комп'ютера наступним вимогам (див. таблицю 4.1):

Таблиця 4.1 – Вимоги до апаратного та програмного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
1	2	3
Процесор	Intel Core i5 або аналогічний	Intel Core i3 або аналогічний
Оперативна пам'ять	4 ГБ RAM	2 ГБ RAM
Вільний простір на диску	500 МБ	100 МБ
Відеокарта	Будь-яка сучасна відеокарта	Будь-яка відеокарта, сумісна з сучасними браузерами
Операційна система	Windows 10 або новіша, macOS, Linux.	Windows 7, macOS, Linux
Браузер	Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.	Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.
Мережа Інтернет	Не потрібна для локального використання після завантаження сторінки.	Не потрібна для локального використання після завантаження сторінки.
Роздільна здатність екрана	HD (1920x1080) або вище	HD (1280x720)
Додаткове обладнання	Миша, клавіатура	Миша, клавіатура

Після успішного завантаження вебсторінки з програмною системою навчання гри на музичних інструментах, користувач потрапляє на головний екран, де представлений весь функціонал програми.

Крок 1. Завантаження аудіофайлу:

1. У першій області з підписом «Завантажити аудіофайл» натисніть кнопку «Обрати файл» або відповідний елемент інтерфейсу для завантаження композиції.

2. У діалоговому вікні виберіть необхідний аудіофайл з підтримуваним форматом та натисніть «Відкрити». Після успішного завантаження, в області відобразиться інформація про нього.

Переконайтеся, що був обраний вірний аудіофайл. При завантаженні композиції, програма виведе діалогове вікно, в якому буде написана інформація про аудіофайл (час, назва та розмір файлу). У випадку завантаження файлу невідповідного формату програма видасть помилку.

Повідомлення про помилку – якщо при завантаженні файлу з'являється повідомлення «Непідтримуваний тип файлу: [тип файлу]», це означає, що завантажений файл має формат, який не підтримується системою.

Крок 2. Очікування та перегляд результатів:

1. Після успішного завантаження аудіофайлу програма автоматично переходить до аналізу та підготовки даних для подальшої роботи. Система спочатку перевіряє цілісність файлу, переконуючись, що він не пошкоджений і придатний для обробки.

2. Після успішного підключення музичного інструменту користувач отримує доступ до розширеного меню налаштувань, яке дозволяє персоналізувати параметри звучання під свої уподобання та технічні особливості обладнання. Цей функціонал реалізований через інтуїтивну панель керування, що включає кілька ключових розділів.

Крок 3. Повторне порівняння або завантаження нових зображень.

Для подальшого навчання, користувач може повторно виконати Крок 1, завантаживши нові аудіофайли. Попередні файли будуть збережені, допоки користувач власноруч не видалить їх.

Дана інструкція охоплює основний функціонал підсистеми навчання гри на музичних інструментах.

4.3 Висновки

У розділ детально описується процес тестування підсистеми навчання гри на музичних інструментах. Було визначено основні стани системи, можливі помилки (непідтримуваний тип файлу, помилка підключення інструментів). Розроблена інструкція користувача надає чітке керівництво щодо технічних вимог та порядку роботи з системою, включаючи завантаження аудіофайлів та налаштування музичних інструментів. Проведене тестування та розроблена інструкція є важливими кроками для забезпечення надійності, зручності використання та розуміння принципів роботи підсистеми навчання гри на музичних інструментах кінцевими користувачами.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було проведено аналіз, проектування, розробку програмних модулів додатку для навчання гри на музичних інструментах. Метою було підвищення ефективності та покращення якості навчання гри на музичних інструментах шляхом розробки програмних модулів. Було використано сучасні фронтенд-інструменти, таких як Java Sound API для роботи з аудіовхідом/виходом, MIDI-пристроями, запису та відтворення звуку, графічний інтерфейс (GUI), а також порівняно Fender Play, Jamorama та TrueFire, щоб визначити їх сильні та слабкі сторони. На основі цього сформульовано вимоги до власного ПЗ.

Створено JavaFX-інтерфейс, інтегровано бібліотеки для створення зручного і зрозумілого інтерфейсу для роботи з програмою для навчання гри на музичних інструментах.

Тестування виявило помилки (формат файлу, відсутність музичного інструменту) та передбачило попередження для користувача. Розроблена інструкція полегшує освоєння системи.

Були виконані поставлені завдання:

- визначити основні вимоги до застосунку для навчання гри на музичних інструментах, з урахуванням особливостей навчального процесу;
- розробити інтерактивну програму для навчання гри на гітарі з інтуїтивним інтерфейсом редагування композицій;
- створити програмні модулі у додатку на основі мови програмування Java;
- розробити адаптивний графічний інтерфейс для зручної взаємодії користувача з навчальною системою;
- забезпечити інструментарій для налаштування тривалості нот, динаміки та спецефектів у процесі створення музичних композицій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. QuData [Електронний ресурс] URL: https://qudata.com/en/ai-ml-case-studies/game-processes-analysis/ (дата звернення: 05.05.2025).

2. Musical Instrument Identification Using Deep Learning Approach [Електронний ресурс] URL: https://www.mdpi.com/1424-8220/22/8/3033 (дата звернення: 08.05.2025).

3. МОДЕЛЬ ВЗАЄМОДІЇ ПРОГРАМ І СИСТЕМ [Електронний ресурс] URL: https://stud.com.ua/164964/informatika/model_vzayemodiyi_program_sistem (дата звернення: 08.05.2025).

4. Заболотчук А. В., Стахов О. Я., Романюк О. Н. «Computer programs for learning to play musical instruments» / Матеріали Міжнародна науково-практична інтернет-конференція студентів аспірантів та молодих науковців (2025) [Електронний ресурс] URL: https:// conferences.vntu.edu.ua/index.php/mn/mn2025/ author/submission/25569

5. MIDI-technology [Електронний ресурс] URL: https://muzline.ua/uk/statti-ta-oglyadi/sintezator-abo-midi-klaviatura--shto-krashte-i-dlya-yakih-tsiley-vikoristovu--tysya/ (дата звернення: 10.05.2025).

6. How AI is changing music education [Електронний ресурс] URL: https://www.unite.ai/uk/ (дата звернення: 10.05.2025).

7. Gonzalez R. C. Digital Image Processing / R. C. Gonzalez, R. E. Woods. – 4th ed. – Pearson, 2022. – 1024 p.

8. Gamification in music education [Електронний ресурс] URL: https://drimify.com/en/resources/gamification-music-education/ (дата звернення: 10.05.2025).

9. Akramullah S. Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis. – Apress, 2024. – 368p.

10. Fender Play [Електронний ресурс] URL: https://www.fender.com/play (дата звернення: 23.05.2025).

11. Jamorama [Електронний ресурс] URL: https://jamorama.com/ (дата звернення: 23.05.2025).

12. TrueFire [Електронний ресурс] URL: https://truefire.com/online-guitar-lessons-cc (дата звернення: 23.05.2025).

13. What is Java [Електронний ресурс] URL: https://goit.global/ua/articles/shcho-take-java-i-de-vona-vykorystovuietsia/ (дата звернення: 24.05.2025).

14. 30 Best Free Tools for Frontend Developers in 2025 [Електронний ресурс] URL: https://dev.to/rowsanali/30-best-free-tools-for-frontend-developers-in-2025-1gdm (дата звернення: 25.05.2025).

15. A. Banks, E. Porcello. Learning React: Modern Patterns for Developing React Apps 2nd Edition. Sebastopol: O'Reilly, 2023. – 307 с.

16. TarsosDSP: a Java Library for Audio Processing [Електронний ресурс] URL: https://0110.be/Software (дата звернення: 25.05.2025).

17. D. Flanagan. JavaScript. The Definitive Guide. Master the World's Most-Used Programming Language 7th Edition. Sebastopol: O'Reilly, 2020. – 706 с.

18. JSyn - Audio Synthesis API for Java [Електронний ресурс] URL: https://www.softsynth.com/jsyn/ (дата звернення: 26.05.2025).

19. G. Blokdyk. System Testing A Complete Guide - 2020 Edition. 5STARCooks, 2021. – 304 с.

ДОДАТКИ

ДОДАТОК А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк _____
«25» березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка додатку для навчання гри на музичних інструментах» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
_____ д. ф., ст. викл. каф. ПЗ О. Я. Стахов
«24» березня 2025 р.

Виконав:
_____ студент гр. 3ПІ-21Б А. В. Заболотчук
«24» березня 2025 р.

Вінниця – 2025

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка додатку для навчання гри на музичних інструментах».

Галузь застосування – додаток для навчання гри на музичних інструментах.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Метою бакалаврської кваліфікаційної роботи є створення програмних модулів для додатку, що сприятимуть підвищенню ефективності навчанню гри на музичних інструментах завдяки інтерактивному контенту, можливості точних налаштувань музичних параметрів та чітких позначень для гітарних технік.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Learn JavaFX 17. Building User Experience and Interfaces with Java. 2nd Ed, 2022.
2. Pro JPA 2. Mastering the Java Persistence API (Expert's Voice in Java Technology), 2023.

5. Технічні вимоги

Вихідні дані до роботи: результати тестування модулів, скріншоти, результат тестування додатку.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до БКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з\п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз розвитку технологій розвитку технологій веб-навчання гри на музичних інструментах та постановка задач	25.03.25 – 04.04.25
2	Розробка моделі та алгоритмів програмного продукту	04.04.25 – 27.04.25
3	Розробка програмних модулів додатку для навчання гри на музичних інструментах	27.04.25 – 12.05.25
4	Тестування програми	12.05.25 – 24.05.25
5	Оформлення матеріалів до захисту БКР	24.05.25 – 30.05.25

10. Порядок контролю та прийняття. Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б. Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка додатку для навчання гри на музичних інструментах»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра ПЗ, ФІТКІ, гр. ЗПІ-21Б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 6.2 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Стахов О. Я. д. ф., ст. викл. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Заболотчук А. В.
(підпис) (прізвище, ініціали)

ДОДАТОК В. Лістинг програми

```
import org.herac.tuxguitar.app.action.impl.system.TGDisposeAction;
import org.herac.tuxguitar.app.system.config.TGConfigKeys;
import org.herac.tuxguitar.app.system.config.TGConfigManager;
import org.herac.tuxguitar.app.system.icons.TGIconEvent;
import org.herac.tuxguitar.app.system.icons.TGIconManager;
import org.herac.tuxguitar.app.ui.TGApplication;
import org.herac.tuxguitar.app.view.component.tabfolder.TGTabFolder;
import org.herac.tuxguitar.app.view.component.table.TGTableViewer;
import org.herac.tuxguitar.app.view.dialog.fretboard.TGFretBoardEditor;
import org.herac.tuxguitar.app.view.toolbar.edit.TGEditToolBar;
import org.herac.tuxguitar.app.view.toolbar.main.TGMainToolBar;
import org.herac.tuxguitar.app.view.util.TGCursorController;
import org.herac.tuxguitar.app.view.util.TGSyncProcess;
import org.herac.tuxguitar.event.TGEvent;
import org.herac.tuxguitar.event.TGEventListener;
import org.herac.tuxguitar.ui.UIFactory;
import org.herac.tuxguitar.ui.layout.UITableLayout;
import org.herac.tuxguitar.ui.resource.UICursor;
import org.herac.tuxguitar.ui.resource.UIRectangle;
import org.herac.tuxguitar.ui.resource.UISize;
import org.herac.tuxguitar.ui.widget.UIPanel;
import org.herac.tuxguitar.ui.widget.UIWindow;
import org.herac.tuxguitar.util.TGContext;
import org.herac.tuxguitar.util.TGExpressionResolver;
import org.herac.tuxguitar.util.TGSynchronizer;
import org.herac.tuxguitar.util.singleton.TGSingletonFactory;
import org.herac.tuxguitar.util.singleton.TGSingletonUtil;

public class TGWindow implements TGEventListener {

	private TGContext context;
	private TGSyncProcess loadTitleProcess;
	private TGCursorController cursorController;

	private UIWindow window;

	public TGWindow(TGContext context) {
		this.context = context;
		this.createSyncProcesses();
	}

	public void open() {
		if( this.window != null ) {
			this.window.open();
			this.window.layout();
		}
	}
```

```
	public void createWindow() {
		UIFactory uiFactory = TGApplication.getInstance(this.context).getFactory();

		this.window = uiFactory.createWindow();
		this.window.addCloseListener(new TGActionProcessorListener(this.context,
TGDisposeAction.NAME));

		this.createShellComposites(uiFactory);
		this.createShellListeners();
		this.loadIcons();
		this.loadInitialBounds();
	}

	private void createShellComposites(UIFactory uiFactory) {
		TGConfigManager tgConfig = TGConfigManager.getInstance(this.context);

		TGMainToolBar tgToolBar = TGMainToolBar.getInstance(this.context);
		tgToolBar.createToolBar(this.window,
tgConfig.getBooleanValue(TGConfigKeys.SHOW_MAIN_TOOLBAR));

		UITableLayout topContainerLayout = new UITableLayout(0f);
		UIPanel topContainer = uiFactory.createPanel(this.window, false);
		topContainer.setLayout(topContainerLayout);
		topContainerLayout.set(UITableLayout.IGNORE_INVISIBLE, true);

		TGEditToolBar tgEditToolBar = TGEditToolBar.getInstance(this.context);
		tgEditToolBar.createToolBar(topContainer,
tgConfig.getBooleanValue(TGConfigKeys.SHOW_EDIT_TOOLBAR));
		topContainerLayout.set(tgEditToolBar.getControl(), 1, 1, UITableLayout.ALIGN_FILL,
UITableLayout.ALIGN_FILL, false, true, 1, 1, null, null, 0f);
		topContainerLayout.set(tgEditToolBar.getControl(),
UITableLayout.PACKED_HEIGHT, 0f);

		TGTabFolder tgTabFolder = TGTabFolder.getInstance(this.context);
		tgTabFolder.init(topContainer);
		topContainerLayout.set(tgTabFolder.getControl(), 1, 2, UITableLayout.ALIGN_FILL,
UITableLayout.ALIGN_FILL, true, true, 1, 1, null, null, 0f);
		topContainerLayout.set(tgTabFolder.getControl(), UITableLayout.PACKED_HEIGHT,
0f);

		TGWindowDivider tgWindowDivider = new TGWindowDivider(this.context);
		tgWindowDivider.createDivider(this.window);

		TGTableViewer tgTableViewer = TGTableViewer.getInstance(this.context);
		tgTableViewer.init(this.window);

		UIPanel bottom = uiFactory.createPanel(this.window, false);
		bottom.setLayout(new UITableLayout(0f));
		bottom.getLayout().set(UITableLayout.IGNORE_INVISIBLE, true);
```

```
		TGFretBoardEditor tgFretBoardEditor = TGFretBoardEditor.getInstance(this.context);
		tgFretBoardEditor.createFretBoard(bottom,
tgConfig.getBooleanValue(TGConfigKeys.SHOW_FRETBOARD));

		// Layout
		this.window.setLayout(new TGWindowLayout(tgToolBar.getControl(), topContainer,
tgWindowDivider.getControl(), tgTableViewer.getControl(), bottom));
	}

	private void loadInitialBounds() {
		TGConfigManager config = TGConfigManager.getInstance(this.context);

		boolean maximized = config.getBooleanValue(TGConfigKeys.MAXIMIZED);
		if( maximized ) {
			this.window.maximize();
		}
		else {
			float width = config.getFloatValue(TGConfigKeys.WIDTH);
			float height = config.getFloatValue(TGConfigKeys.HEIGHT);
			if( width > 0 && height > 0 ){
				UIRectangle uiRectangle = new UIRectangle();
				uiRectangle.setSize(new UISize(width, height));

				this.window.setBounds(uiRectangle);
			}
		}
	}

	private void createShellListeners() {
		TGIconManager tgIconManager = TGIconManager.getInstance(this.context);
		tgIconManager.addLoader(this);
	}

	public void loadDefaultCursor() {
		this.loadCursor(UICursor.NORMAL);
	}

	public void loadBusyCursor() {
		this.loadCursor(UICursor.WAIT);
	}

	public void loadCursor(UICursor cursor) {
		if(!this.isDisposed()) {
			if( this.cursorController == null ||
!this.cursorController.isControlling(this.getWindow()) ) {
				this.cursorController = new TGCursorController(this.context,
this.getWindow());
			}
			this.cursorController.loadCursor(cursor);
		}
	}
```

```
	public void moveToTop() {
		if(!this.isDisposed()) {
			this.getWindow().moveToTop();
		}
	}

	public boolean isDisposed() {
		return (this.getWindow() == null || this.getWindow().isDisposed());
	}

	public UIWindow getWindow() {
		return this.window;
	}

	public void loadTitle() {
		this.loadTitleProcess.process();
	}

	public void loadTitleInCurrentThread() {
		if(!this.isDisposed()) {
			String titleLayout = TGConfigManager.getInstance(this.context).getStringValue(TGConfigKeys.WINDOW_TITLE);
			String title = TGExpressionResolver.getInstance(this.context).resolve(titleLayout);

			this.window.setText(title != null ? title : TGApplication.NAME);
		}
	}

	public void loadIcons() {
		if(!this.isDisposed()) {

	this.getWindow().setImage(TGIconManager.getInstance(this.context).getAppIcon());
			this.getWindow().layout();
		}
	}

	public void createSyncProcesses() {
		this.loadTitleProcess = new TGSyncProcess(this.context, new Runnable() {
			public void run() {
				TGWindow.this.loadTitleInCurrentThread();
			}
		});
	}

	public void processEvent(final TGEvent event) {
		TGSynchronizer.getInstance(this.context).executeLater(new Runnable() {
			public void run() {
				if( TGIconEvent.EVENT_TYPE.equals(event.getEventType()) ) {
					loadIcons();
```

```
				}
			}
		});
	}

	public static TGWindow getInstance(TGContext context) {
		return TGSingletonUtil.getInstance(context, TGWindow.class.getName(), new TGSingletonFactory<TGWindow>() {
			public TGWindow createInstance(TGContext context) {
				return new TGWindow(context);
			}
		});
	}
}

package org.herac.tuxguitar.app.view.main;

import org.herac.tuxguitar.app.ui.TGApplication;
import org.herac.tuxguitar.app.view.component.table.TGTableViewer;
import org.herac.tuxguitar.ui.UIFactory;
import org.herac.tuxguitar.ui.event.UIMouseDragListener;
import org.herac.tuxguitar.ui.event.UIMouseEvent;
import org.herac.tuxguitar.ui.layout.UITableLayout;
import org.herac.tuxguitar.ui.resource.UICursor;
import org.herac.tuxguitar.ui.widget.UIContainer;
import org.herac.tuxguitar.ui.widget.UIControl;
import org.herac.tuxguitar.ui.widget.UIDivider;
import org.herac.tuxguitar.util.TGContext;

public class TGWindowDivider implements UIMouseDragListener {

	private TGContext context;
	private UIDivider divider;

	public TGWindowDivider(TGContext context) {
		this.context = context;
	}

	public void createDivider(UIContainer parent) {
		UIFactory uiFactory = TGApplication.getInstance(this.context).getFactory();

		this.divider = uiFactory.createHorizontalDivider(parent);
		this.divider.addMouseDragListener(this);
		this.divider.setCursor(UICursor.SIZENS);
	}

	public void onMouseDrag(UIMouseEvent event) {
		UIControl control = TGTableViewer.getInstance(this.context).getControl();

		TGWindow tgWindow = TGWindow.getInstance(this.context);
```

```
                TGWindowLayout tgWindowLayout = (TGWindowLayout) tgWindow.getWindow().getLayout();
                tgWindowLayout.set(control, UITableLayout.PACKED_HEIGHT, this.computeHeight(control, event.getPosition().getY()));
                tgWindow.getWindow().layout();
        }

        public Float computeHeight(UIControl control, float move) {
                return Math.max((control.getBounds().getHeight() - move), 0f);
        }

        public UIDivider getControl() {
                return divider;
        }
}

package org.herac.tuxguitar.app.view.main;

import org.herac.tuxguitar.ui.layout.UITableLayout;
import org.herac.tuxguitar.ui.resource.UIRectangle;
import org.herac.tuxguitar.ui.resource.UISize;
import org.herac.tuxguitar.ui.widget.UIControl;
import org.herac.tuxguitar.ui.widget.UILayoutContainer;

public class TGWindowLayout extends UITableLayout {

        private UIControl top;
        private UIControl topContainer;
        private UIControl divider;
        private UIControl bottomContainer;
        private UIControl bottom;

        public TGWindowLayout(UIControl top, UIControl topContainer, UIControl divider, UIControl bottomContainer, UIControl bottom) {
                this.top = top;
                this.topContainer = topContainer;
                this.divider = divider;
                this.bottomContainer = bottomContainer;
                this.bottom = bottom;
                this.configure();
        }

        public UISize computePackedSize(UILayoutContainer container) {
                return this.computePackedSize(container, true);
        }

        public UISize computePackedSize(UILayoutContainer container, boolean resetTableHeight) {
                if( resetTableHeight ) {
                        this.set(this.bottomContainer, MAXIMUM_PACKED_HEIGHT, null);
                }
                return super.computePackedSize(container);
```

```
	}

	public void setBounds(UILayoutContainer container, UIRectangle bounds) {
		UISize packedContentSize = container.getPackedContentSize();
		if( packedContentSize.getHeight() > bounds.getHeight() ) {
			UISize preferredSize = this.getPreferredSizeFor(this.bottomContainer);
			this.set(this.bottomContainer, MAXIMUM_PACKED_HEIGHT,
(preferredSize.getHeight() - (packedContentSize.getHeight() - bounds.getHeight())));
			this.computePackedSize(container, false);
		}

		super.setBounds(container, bounds);
	}

	public void configure() {
		this.set(UITableLayout.MARGIN_TOP, 0f);
		this.set(UITableLayout.IGNORE_INVISIBLE, true);
		this.set(this.top, 1, 1, UITableLayout.ALIGN_FILL, UITableLayout.ALIGN_TOP, true,
false, 1, 1, null, null, 0f);
		this.set(this.topContainer, 2, 1, UITableLayout.ALIGN_FILL,
UITableLayout.ALIGN_FILL, true, true, 1, 1, null, null, 0f);
		this.set(this.topContainer, UITableLayout.PACKED_HEIGHT, 0f);
		this.set(this.divider, 3, 1, UITableLayout.ALIGN_FILL, UITableLayout.ALIGN_FILL,
false, false, 1, 1, null, null, 0f);
		this.set(this.divider, UITableLayout.PACKED_HEIGHT, 4f);
		this.set(this.divider, UITableLayout.MARGIN, 0f);
		this.set(this.bottomContainer, 4, 1, UITableLayout.ALIGN_FILL,
UITableLayout.ALIGN_FILL, true, false, 1, 1, null, null, 0f);
		this.set(this.bottom, 5, 1, UITableLayout.ALIGN_FILL, UITableLayout.ALIGN_FILL,
true, false, 1, 1, null, null, 0f);
	}
}

package org.herac.tuxguitar.app.view.component.tab.edit;

import org.herac.tuxguitar.app.TuxGuitar;
import org.herac.tuxguitar.app.action.impl.edit.tablature.TGMenuShownAction;
import org.herac.tuxguitar.app.action.impl.edit.tablature.TGMouseClickAction;
import org.herac.tuxguitar.app.action.impl.edit.tablature.TGMouseExitAction;
import org.herac.tuxguitar.app.action.impl.edit.tablature.TGMouseMoveAction;
import org.herac.tuxguitar.app.action.impl.layout.TGSetLayoutScaleDecrementAction;
import org.herac.tuxguitar.app.action.impl.layout.TGSetLayoutScaleIncrementAction;
import org.herac.tuxguitar.app.action.listener.gui.TGActionProcessingListener;
import org.herac.tuxguitar.editor.TGEditorManager;
import org.herac.tuxguitar.editor.action.TGActionProcessor;
import org.herac.tuxguitar.player.base.MidiPlayer;
import org.herac.tuxguitar.ui.event.UIMenuEvent;
import org.herac.tuxguitar.ui.event.UIMenuHideListener;
import org.herac.tuxguitar.ui.event.UIMenuShowListener;
import org.herac.tuxguitar.ui.event.UIMouseDownListener;
import org.herac.tuxguitar.ui.event.UIMouseEvent;
```

```
import org.herac.tuxguitar.ui.event.UIMouseExitListener;
import org.herac.tuxguitar.ui.event.UIMouseMoveListener;
import org.herac.tuxguitar.ui.event.UIMouseUpListener;
import org.herac.tuxguitar.ui.event.UIZoomEvent;
import org.herac.tuxguitar.ui.event.UIZoomListener;
import org.herac.tuxguitar.ui.resource.UIPosition;
import org.herac.tuxguitar.util.TGContext;

public class MouseKit implements UIMouseDownListener, UIMouseUpListener,
UIMouseMoveListener, UIMouseExitListener, UIMenuShowListener, UIMenuHideListener,
UIZoomListener {

	private EditorKit kit;
	private UIPosition position;
	private boolean menuOpen;

	public MouseKit(EditorKit kit){
		this.kit = kit;
		this.position = new UIPosition(0,0);
		this.menuOpen = false;
	}

	public boolean isBusy() {
		TGContext context = this.kit.getTablature().getContext();
		return (TGEditorManager.getInstance(context).isLocked() ||
MidiPlayer.getInstance(context).isRunning());
	}

	public void executeAction(String actionId, float x, float y, boolean byPassProcessing) {
		TGActionProcessor tgActionProcessor = new
TGActionProcessor(this.kit.getTablature().getContext(), actionId);
		tgActionProcessor.setAttribute(EditorKit.ATTRIBUTE_X, x);
		tgActionProcessor.setAttribute(EditorKit.ATTRIBUTE_Y, y);

	tgActionProcessor.setAttribute(TGActionProcessingListener.ATTRIBUTE_BY_PASS,
byPassProcessing);
		tgActionProcessor.process();
	}

	public void onMouseDown(UIMouseEvent event) {
		this.position.setX(event.getPosition().getX());
		this.position.setY(event.getPosition().getY());
	}

	public void onMouseUp(UIMouseEvent event) {
		this.position.setX(event.getPosition().getX());
		this.position.setY(event.getPosition().getY());
		this.executeAction(TGMouseClickAction.NAME, event.getPosition().getX(),
event.getPosition().getY(), false);
	}
```

```
		public void onMouseMove(UIMouseEvent event) {
			if(!this.menuOpen && this.kit.isMouseEditionAvailable() && !this.isBusy()){
				this.executeAction(TGMouseMoveAction.NAME,
event.getPosition().getX(), event.getPosition().getY(), true);
			}
		}

		public void onMouseExit(UIMouseEvent event) {
			if(!this.menuOpen && this.kit.isMouseEditionAvailable()) {
				this.executeAction(TGMouseExitAction.NAME,
event.getPosition().getX(), event.getPosition().getY(), true);
			}
		}

		public void onMenuShow(UIMenuEvent event) {
			this.menuOpen = true;
			this.executeAction(TGMenuShownAction.NAME, this.position.getX(),
this.position.getY(), false);
		}

		public void onMenuHide(UIMenuEvent event) {
			this.menuOpen = false;
			TuxGuitar.getInstance().updateCache(true);
		}

		public void onZoom(UIZoomEvent event) {
			if( event.getValue() > 0 ) {
				new TGActionProcessor(this.kit.getTablature().getContext(),
TGSetLayoutScaleIncrementAction.NAME).process();
			} else {
				new TGActionProcessor(this.kit.getTablature().getContext(),
TGSetLayoutScaleDecrementAction.NAME).process();
			}
		}
}
```