**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**„Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет прикладної математики**

“ЗАТВЕРДЖЕНО”

Керівник роботи

\_\_\_\_\_\_\_\_\_\_ Є.С.Сулема

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

**ВЕБ-** **ЗАСТОСУНОК ДЛЯ СТВОРЕННЯ МЕЛОДІЙ ЗА ДОПОМОГОЮ ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ**

**Пояснювальна записка**

ПЗКС.045440-03-81

|  |  |
| --- | --- |
| \_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2011 р. | Виконавець:  \_\_\_\_\_\_\_\_\_\_ Мельничук О.Г. |

2021

**ЗМІСТ**

[СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 3](#_heading=h.gjdgxs)

[ВСТУП 4](#_heading=h.30j0zll)

[1. АНАЛІЗ МОВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ 5](#_heading=h.1fob9te)

[1.1 Мова програмування JavaScript 5](#_heading=h.3znysh7)

[1.2 Cередовище оточення NodeJS 6](#_heading=h.2et92p0)

[1.3 Опис фреймворку ReactJS 7](#_heading=h.tyjcwt)

[1.4 Опис фреймворку ExpressJS 8](#_heading=h.3dy6vkm)

[2. РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ СТВОРЕННЯ МЕЛОДІЙ ЗА ДОПОМОГОЮ ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ 9](#_heading=h.1t3h5sf)

[2.1 Аналіз вимог до розробки веб-застосунку 9](#_heading=h.4d34og8)

[2.2 Опис вимог до розробки веб-застосунку 10](#_heading=h.2s8eyo1)

[2.3 Архітектура веб-застосунку 11](#_heading=h.17dp8vu)

[2.4 Вміст та дизайн сторінок веб-застосунку 12](#_heading=h.3rdcrjn)

[3. ОПИС РОЗРОБЛЕНИХ АЛГОРИТМІВ ТА ПІДПРОГРАМ 15](#_heading=h.26in1rg)

[3.1 Особливості реалізації модулю сервера 15](#_heading=h.lnxbz9)

[3.2 Особливості реалізації модулю клієнтського інтерфейсу 16](#_heading=h.35nkun2)

[АНАЛІЗ РОЗРОБЛЕНОГО WEB-РЕСУРСУ 17](#_heading=h.1ksv4uv)

[4.1 Процес тестування 17](#_heading=h.44sinio)

[4.2 Рекомендації щодо подальшого вдосконалення 18](#_heading=h.2jxsxqh)

[ВИСНОВКИ 19](#_heading=h.z337ya)

[СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 20](#_heading=h.3j2qqm3)

**СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ**

**HTTP** (Hyper Text Transfer Protocol) – протокол передачі гіпертексту;

**API** (Application Programming Interface) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення;

**HTML** (HyperText Markup Language) – мова розмітки документів гіпертексту, складається з тегів;

**JS** (JavaScript) – високорівнева мультипарадигмальна мова програмування, що написана за стандартом ECMAScript;

**FFmpeg** – набір бібліотек для роботи з різними видами мультимедія.

`

**ВСТУП**

Музика вважається однією з найстаріших та найбільш універсальних форм мистецтва. Мистецтво супроводжувало людство усім щаблях його розвитку та приймало яких-завгодно форм. І з них музика – вид творчості, що виник у часи, котрий зазнав величезні зміни засобів і видів, і справедливо вважається однією з улюблених і значимих в людини.

Кожна частина світу має свою окрему, аутентичну музичну культуру. Музична теорія однієї частини світу може дуже сильно відрізнятись від музичної теорії іншої. Невід’ємною частиною будь-якої музичної культури є інструменти. Музичні інструменти створювались та вдосконалювались протягом століть, проте не всюди вони доступні.

Створення веб-застосунку віртуальних інструментів є цікавою та актуальною задачею. Ці інструменти є шансом спробувати щось нове і незвичайне замість звичайного фортепіано та дізнатись трохи про культуру з іншого куточка світу.

Дана курсова робота присвячена розробленню веб-застосунку набору віртуальних інструментів, який дасть можливість спробувати зіграти та створити композиції на екзотичних музикальних інструментах.

**1. АНАЛІЗ МОВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ**

**РОЗРОБЛЕННЯ WEB-САЙТІВ**

* 1. **Мова програмування JavaScript**

**JavaScript** — мультипарадигмова мова програмування, що є реалізацією стандарту ECMAScript. Підтримує об’єктно-орієнтований, функціональний та імперативні парадигми програмування. Використовується переважно у веб-програмуванні для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

JavaScript класифікують як прототипну скриптову мову програмування з динамічною типізацією. JavaScript є мультипарадигмовою, що означає те що вона підтримує декілька парадигм программування і має деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

JavaScript це мова з великими перевагами, які роблять його найкращим вибором серед подібних йому, особливо в деяких варіантах застосування. Кілька переваг використання JavaScript:

* JavaScript є інтерпретованою мовою: веб-браузер сам швидко інтерпретує код в HTML і вам не потрібний ніякий компілятор;
* JavaScript порівняно простіша ніж інші мови програмування;
* Помилки доволі просто виявити;
* JavaScript дуже розповсюджена мова, має велику кількість навчальних матеріалів та бібліотек;
* JS працює в різних браузерах і на різних платформах.
  1. **Середовище оточення NodeJS**

**NodeJS** або **Node.js** — кросплатформове середовище виконання JavaScript з відкритим кодом, що перетворює JavaScript у мову загального призначення та побудоване на основі двигуна JavaScript Chrome V8. Node.js дозволяє взаємодіяти з пристроями вводу-виводу через власний API та підключати бібліотеки інших мов програмування.

Node часто використовується у якості веб-сервера і спроектований для побудови масштабованих мережевих додатків. Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає цикл подій (event loop) за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

Зазвичай поведінка визначається через функції зворотнього виклику на початку скрипта і в кінці запускає сервер через блокуючий виклик. В Node.js немає нічого подібного на виклик початку циклу подій. Node.js просто входить в подієвий цикл після запуску скрипта на виконання. Node.js виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотнього виклику. Така поведінка схожа на поведінку браузерного JavaScript: подієвий цикл прихований від користувача.

HTTP є об'єктом першого роду в Node.js, розробленим з потоковістю та малою затримкою. Це робить Node.js хорошою основою для веб-бібліотеки або фреймворку.

* 1. **Опис фреймворку ReactJS**

**ReactJS** (або просто **React**) — JavaScript бібліотека з відкритим кодом для створення інтеркативних інтерфейсів. Ця бібліотека покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. React вирішує ці проблеми дозволяючм розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки.

React заснована на «компонентах» - інкапсульованих частинах інтерфейсу з власним станом, що замінюють шаблони та дозволяють зберігати дані та стан окремо від DOM. Кожен компонент повертає частину призначеного для користувача інтерфейсу зі своїм станом та даними. Об'єднуючи компоненти, програміст створює завершений інтерфейс веб-додатки.

За допомогою React розробник описує, як компоненти інтерфейсу виглядають в різних станах. Декларативний підхід скорочує код і робить його зрозумілим. Компоненти реалізують метод render(), який приймає вхідні дані і повертає те, що буде показано користувачу. У методі часто використовується XML-подібний синтаксис під назвою JSX.

React можна використовувати на серверах за допомогою Node, а також на мобільних платформах за допомогою React Native.

* 1. **Опис фреймворку ExpressJS**

**ExpressJS** (або просто **Express**) – легка модульна веб-структура NodeJS для написання веб- та мобільних застосунків. Express надає широкий вибір службових методів HTTP та проміжних обробників, що допомагає створити якісний та надійний API для сервера.

Філософія Express полягає в наданні невеликих, надійних інструментів для HTTP-серверів, що робить його чудовим рішенням для односторінкових програм, веб-сайтів, гібридів або загальнодоступних API HTTP.

Express не змушує використовувати будь-який конкретний ORM або механізм шаблонів. Завдяки підтримці понад 14 механізмів шаблонів через Consolidate.js є можливість швидко створити свій ідеальний фреймворк.

Express було створено в Node.js, тому він успадковує його безліч функцій, таких як гнучкість, масштабованість, простота, мінімалізм, а також продуктивність. Express набув велику популярність і дуже часто йде пліч-о-пліч з Node.js при написанні веб-застосунків.

**2. РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ СТВОРЕННЯ МЕЛОДІЙ ЗА ДОПОМОГОЮ ВІРТУАЛЬНИХ ІНСТРУМЕНТІВ**

## **2.1 Аналіз вимог до розробки веб-сервісу**

Аналіз вимог – частина процесу розробки програмного забезпечення. Включає в себе збір вимог до програмного забезпечення, їх систематизацію, виявлення взаємозв'язків, а також створення документації. В процесі збору вимог важливо брати до уваги можливі протиріччя вимог різних зацікавлених осіб, таких як замовники, так і для розробників.

В якості розробника виступають студенти, в якості замовника – викладач.

Процес аналізу вимог до розробки веб-сервісу складається з трьох видів діяльності:

* збір вимог – спілкування з клієнтами і користувачами, щоб визначити, які їхні вимоги, а також аналіз предметної області;
* аналіз вимог – визначення, чи є зібрані вимоги неясними, неповними, неоднозначними або такими, що суперечать. А також рішення проблем та виявлення взаємозв'язку вимог;
* документування вимог – вимоги можуть бути задокументовані в різних формах, таких як простий опис, сценарії використання, користувацькі історії, або специфікації процесів.

Таким чином, були створені вимоги до розробки та функціональності веб-додатку створення мелодій за допомогою віртуального піаніно, сформовані та описані під час етапу документування вимог.

**2.2 Опис вимог до розробки веб-сервісу**

Кожна вимога має власний індивідуальний код. Усі вимоги зазначені у даній таблиці:

Таблиця 1. Опис вимог до проекту

|  |  |
| --- | --- |
| Код вимоги | Опис вимоги |
| А1 | Кожний інструмент має власний дизайн та ноти |
| А2 | Підсвічування клавіш інструмента при натисканні клавіши |
| А3 | Можливість одночасного натискання декількох клавіш |
| А4 | Можливість натискання клавіши за допомогою клавіатури |
| А5 | Деякі інструменти мають додаткові набори нот які можна змінити |
| А6 | Можливість прослуховування записаної мелодії |
| А7 | Можливість завантаження на пристрій записаної мелодії |

## **2.3 Архітектура веб-додатку**

Даний веб-додаток складається з таких модулів:

* Модуля серверу
* Модуля клієнтського інтерфейсу
  + Модуля інструмента
  + Модуля панелі керування (кнопок)

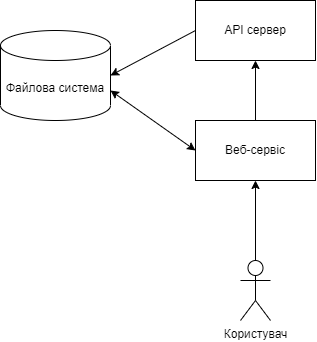


Рис. 1 Архітектура системи

Модуль серверу призначений для створення програми HTTP-серверу, який дозволяє клієнтській стороні взаємодіяти з файловою системою через його API. Модуль призначений для генерації аудіофайлу записаної мелодії користувача.

Модуль клієнтського інтерфейсу представляє собою односторінковий веб-сайт, який дозволяє користувачу через елементи керування на ньому створювати, прослуховувати та завантажувати зіграні мелодії. Цей модуль безпосередньо взаємодіє із модулем серверу через його API.

## **2.4 Вміст та дизайн сторінок веб-застосунку**

Даний веб-додаток являє собою HTML сторінку, на якій відображено панель керування з кнопками та клавіші вибраного інструмента.

Віртуальний інструмент складається лише з однієї сторінки, вся логіка взаємодії з якою прописана за допомогою JS коду.

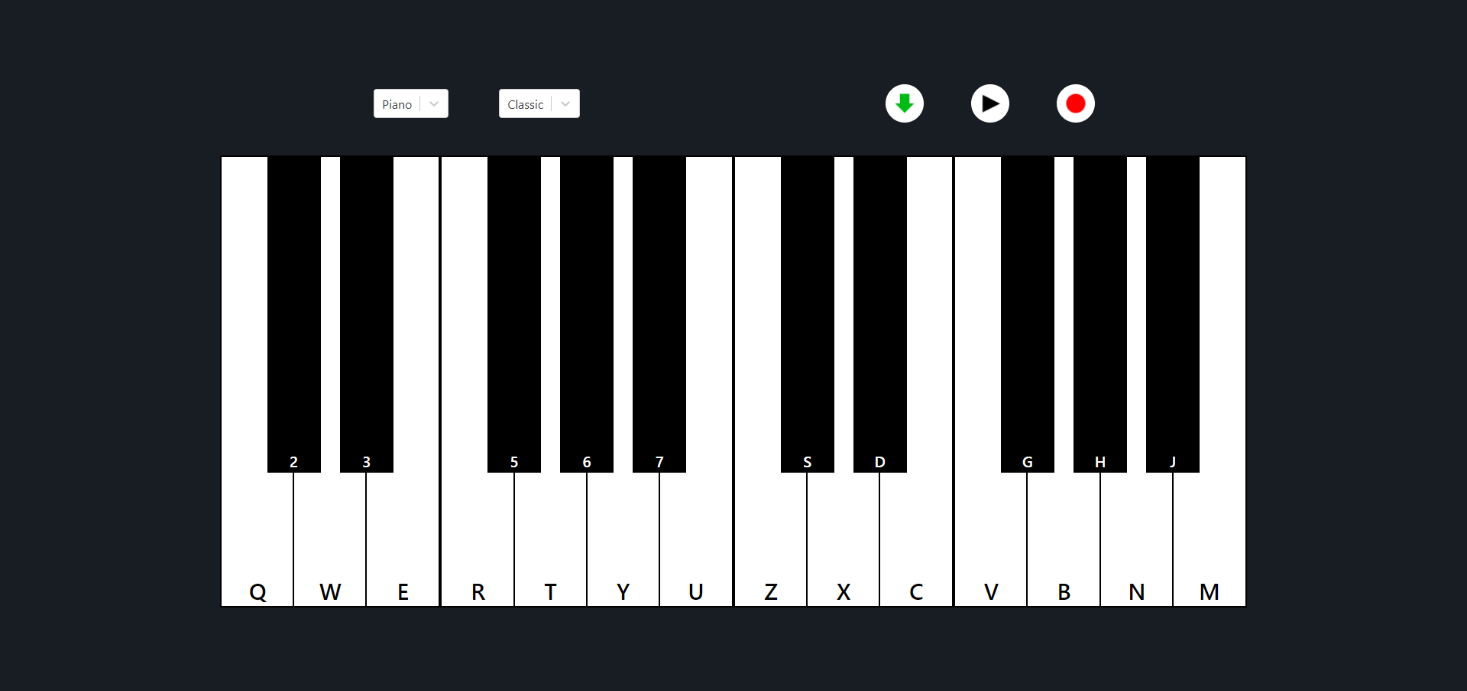
На панелі керування розташовані 3 кнопки керування зіграними мелодії, а саме кнопка початку запису мелодії, кнопка програвання записаної мелодії та кнопка завантаження програної мелодїі.

Рис. 2 Сторінка браузерного віртуального інструмента: піаніно

На скріншоті відображена головна сторінка додатку. Додаток складається з панелі кнопок та панелі інструмента. Панель кнопок має такі кнопки:

* Кнопка з вибором інструмента зі списку
* Кнопка з вибором налаштування (tuning) обраного інструмента зі списку
* Кнопка завантаження записаної мелодії
* Кнопка програвання записаної мелодії
* Кнопка включення/виключення запису мелодії

Кнопки з вибором інструмента/нот інструмента є спадними списками.

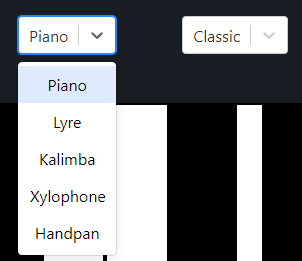
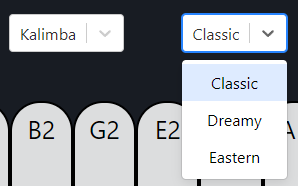
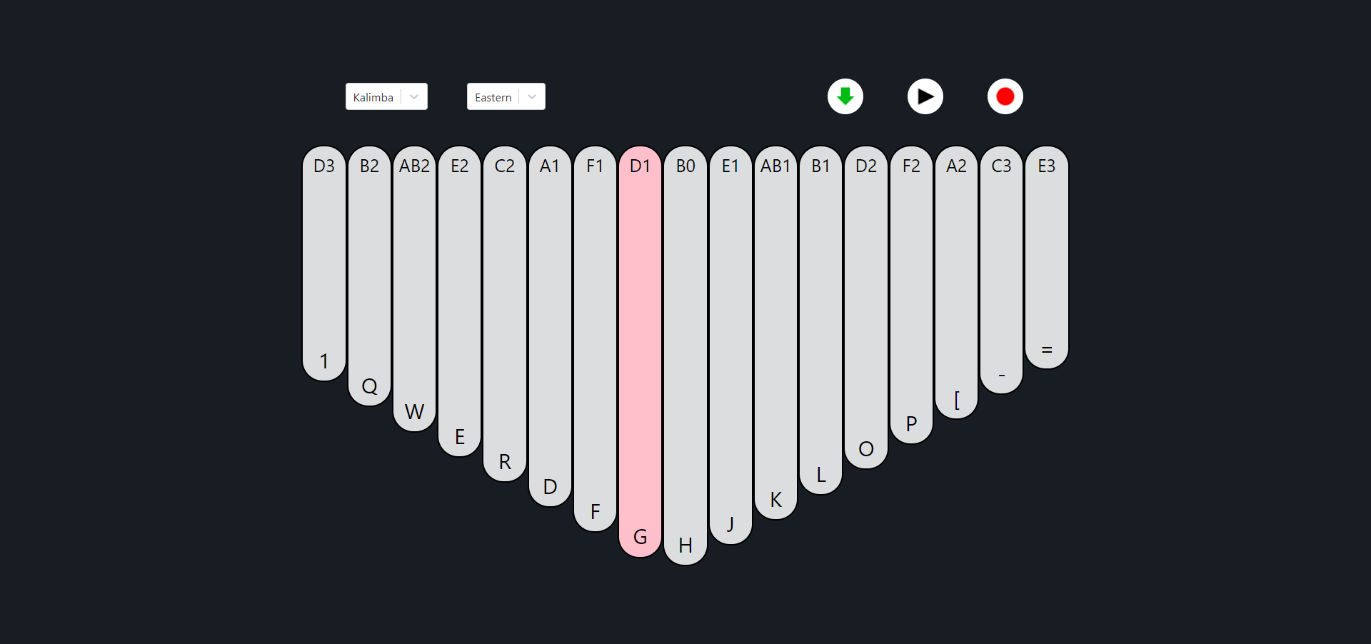
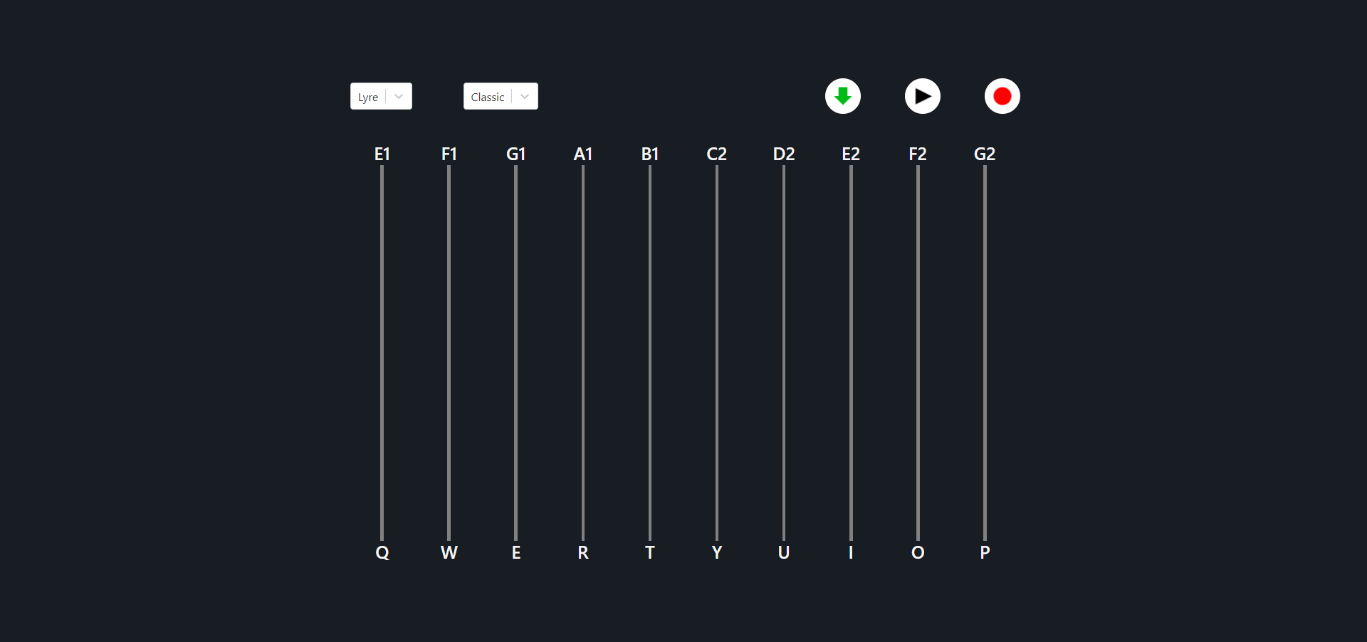


Рис. 3.1-2 Приклади списків інструментів/тюнингів

Кнопки вибору інструмента завантажують новий інструмент замість старого. Кожний інструмент має свій візуальний дизайн та свої кнопки.

Кнопки вибору нот підключають інші звуки нот на клавіші.

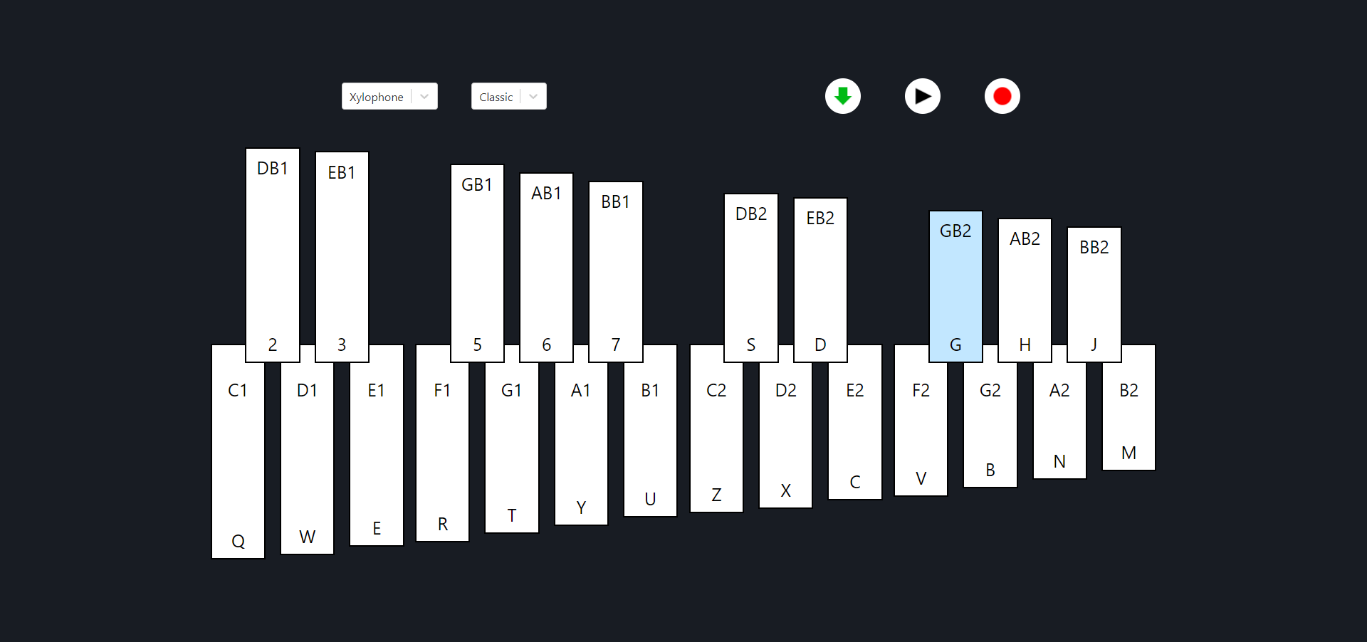
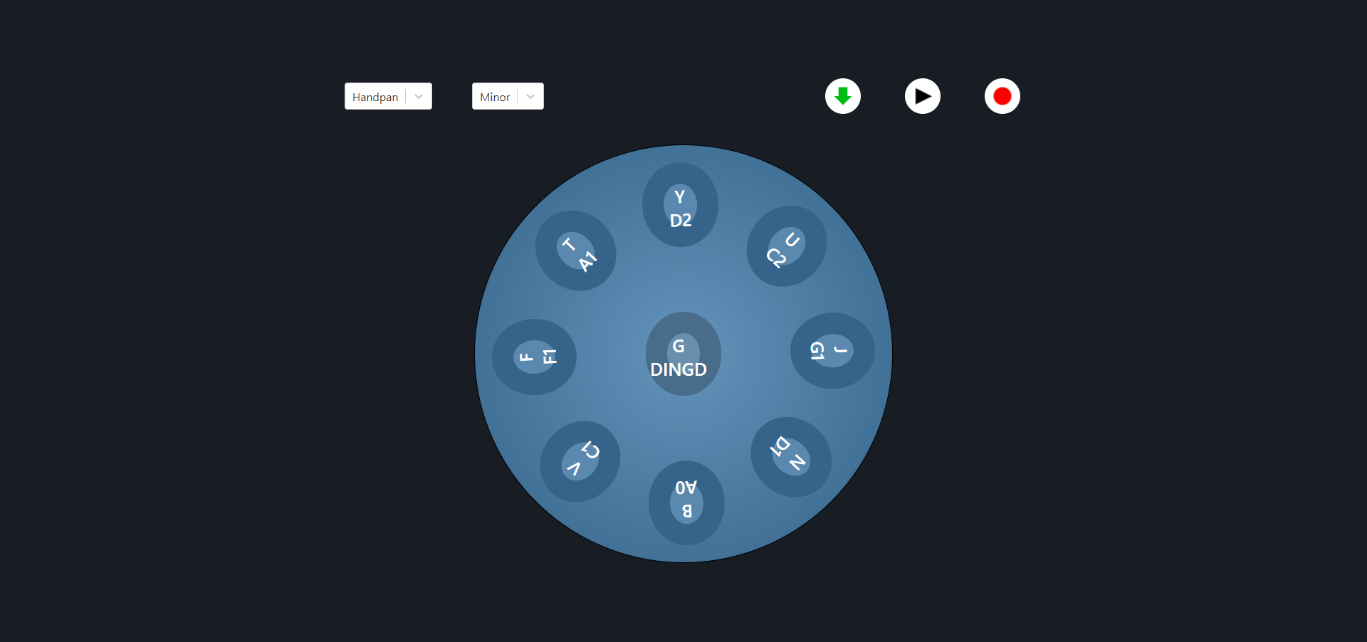


Рис. 4.1-4 Візуальний вигляд інструментів: калімба, арфа, глокеншпіль, ханг

**3. ОПИС РОЗРОБЛЕНИХ АЛГОРИТМІВ ТА ПІДПРОГРАМ**

**3.1 Особливості реалізації модулю сервера**

Модуль серверу було створено на мові програмування JavaScript у середовищі Node.js із використанням фреймворку Express.

Даний модуль підтримує взаємодію із клієнтською частиною через API побудоване за моделлю REST. Він дозволяє створювати аудіо файл на основі програних користувачем нот на віртуальному інструменті.

Для створення аудіо файлу потрібно надіслати HTTP запит на сервер із JSON-об’єктом, що містить список програних нот, який утворився при натисканні кнопки «Записати» на клієнтській частині системи, та шлях до нот на базі даних сервера. За допомогою списку програних нот сервер буде співставлювати назву ноти, що була зіграна в певний момент часу, з фізичним представленням ноти на пристрої, де знаходиться сервер. Зі списку нот складається команда, яка подається до бібліотеки **Ffmpeg**, що в свою чергу створює аудіо файл, що надсилається у відповідь на HTTP запит. Аудіо файл зберігається на стороні серверу з назвою «recording» у форматі .mp3.

**3.2 Особливості реалізації модулю клієнтського інтерфейсу**

Модуль клієнтського інтерфейсу було створено з використанням мови програмування JavaScript та фреймворку ReactJS у вигляді веб-додатку, який взаємодіє із API сервером. Користувач має можливість натискати на клавіші віртуального інструмента та створювати мелодії. Також користувач може записати свою створену мелодію та прослухати її, разом з цим він також може зберегти записані мелодії на пристрій для особистого користування використовуючи панель керування, що складається з 3 кнопок та 2 кнопок-списків.

При натисканні кнопки запису всі звуки від клавіш, що натискаються користувачем записуються в список, який містить назву ноти, що була зіграна, та час, коли була натиснута відповідна клавіша.

При натисканні кнопки «Грати» список програних нот відтворюється за допомогою емуляції натискання клавіш користувачем. Тобто, відтворюється програна користувачем мелодія на піаніно з підсвіткою клавіш, для розуміння, яка клавіша була натиснута в певний момент.

При натисканні кнопки «Завантажити» надсилається запит на API серверу для створення аудіо файлу. Запит містить JSON-об’єкт з даними для створення файлу, а саме раніше зазначений список програних нот, що утворився при натисканні кнопки «Записати», та інструмент з якого було зіграно ноти. У відповідь користувач отримує посилання на файл, який ожна завантажити на пристрій та прослухати.

# **АНАЛІЗ РОЗРОБЛЕНОГО WEB-РЕСУРСУ**

## **4.1 Процес тестування**

Тестування системи відбулося шляхом ручного тестування.

**Ручне тестування** (Manual testing) — процес ручної перевірки програмного забезпечення на помилки. Тестувальник відіграває роль користувача програми й використовує властивості програми для знаходження помилок у роботі програми. Для професійного тестування тестувальник часто користується написаним планом тестування з варіантами тестування (test cases).

Для тестування було написано та протестовані такі варіанти тестування

* Запис пустої мелодії
* Програвання пустої мелодії
* Запис мелодії з одночасним натисканням кількох клавіш
* Запис мелодії на різних налаштуваннях інструментів

А також стандартні варіанти такі як:

* Натискання клавіш клавіатурою
* Запис мелодії
* Програвання записаної мелодії
* Завантаження записаної мелодії

Усі варіанти тестування були успішно виконаними.

**4.2 Рекомендації щодо подальшого вдосконалення**

В процесі роботи над веб-додатком було виявлено ряд напрямів та функцій, які можуть покращити розроблене ПЗ, а саме:

* Створити редактор нот, де програні ноти можуть бути записані та експортовані у різних текстових форматах та .midi
* Пововнити базу даних з нотами для того щоб можна було створити більше налаштувань для інструментів
* Додати вибір різних кольорових тем
* Додати різні налаштування відтворення записаних мелодій, наприклад гучність та швидкість

**ВИСНОВКИ**

Метою даного курсового проекту було розроблення веб-застосунку для створення мелодій за допомогою віртуальних інструментів.

Розроблений веб-додаток дозволяє:

* Програвання нот за допомогою клавіатури
* Запис мелодій
* Прослуховування створеної мелодії
* Завантаження створеної мелодії на пристрій

Розробка виконана у повному обсязі, всі вимоги, викладені у технічному завданні враховані, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

Використання цього веб-застосунку дозволить користувачам власноруч створювати композиції за допомогою віртуальних інструментів.

**СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ**

1. Ручне тестування [Електронний ресурс]. — Режим доступу :

[https://uk.wikipedia.org/wiki/Ручне\_тестування](https://uk.wikipedia.org/wiki/%D0%A0%D1%83%D1%87%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)

1. ReactJS Documentation [Електронний ресурс]. — https://reactjs.org/docs/getting-started.html
2. Express: API Reference [Електронний ресурс]. — <https://expressjs.com/en/5x/api.html>
3. FFmpeg Documentation [Електронний ресурс]. http://www.ffmpeg.org/documentation.html
4. Аналіз вимог [Електронний ресурс]. — Режим доступу: <https://habr.com/post/340956/>