Development of an Intelligent System for Selecting Songs According to the User Needs

Roman Kubik[1], Yuriy Ryshkovets[0000-0001-5831-4000]2, Mariya Hrendus[0000-0003-3832-4716]3, Andrii Khudyi[0000-0003-2029-7270]4, Anatolii Vysotskyi[5], Viktor Hryhorovych[0000-0002-5828-067X]6, Sofiia Chyrun[0000-0002-2829-0164]7

[1-4]Lviv Polytechnic National University, Lviv, Ukraine
[5]Anat Company, Lviv, Ukraine
[6]Drohobych Ivan Franko State Pedagogical University, Drohobych, Ukraine
[7]IT Step University, Lviv, Ukraine

roman89kubik@gmail.com[1], Yuriy.V.Ryshkovets@lpnu.ua[2], mariya.h.hirnyak@lpnu.ua[3], Khudyy@ukr.net[4], anat1957@gmail.com[5], viktor.grigorovich@gmail.com[6], chyrunsofia@gmail.com[7]

Abstract. The purpose of the work was to create an intelligent system for the selection of songs following the user needs. The song is something that all people like. It might be cheerful and sad. The song can describe both the happiest moments of a person's life and tragic experiences. The song accompanies a person from his/her birth, when the mother sings lullabies to the child, until death, namely funeral songs. However, in most cases, songs dominate in our present, and especially during the meeting of friends or people with common hobbies in an informal atmosphere. In order not to stand aside among the crowd of people who sing a particular song, you need to be able to sing it. There are many sources in access to get this or that lyrics. The first one is the Internet. You can also find the lyrics in a special printed collection of songs. However, often you can be out of the coverage, or there is no Internet connection.

Keywords. Language, Vocabulary, Intelligent System, Songs, Data Aggregation, Content, Intelligent System, Content Analysis, Data Mining, Context Filtering, Online Learning System, User Needs

1 Introduction

The paper directory partially solves this problem, but the other one is that you need to constantly carry this directory. Of course, a person who is not a frequent participant in this type of event will not carry it around all day. However, in the era of digital technology and mobile phones, it is convenient to have such a collection on your mobile device. This is the kind of application that will be developed, as a work. Its main difference from many other similar applications will be the ability to analyze the mood of the user, and accordingly, give a hint about the next song [1-7]. This topic is extremely relevant, because themed nights, where young people gather and sing songs

are becoming more popular every day. In addition, a considerable audience of potential users of this intelligent system is Plast members, who regularly organize Plast campfires, on which it is customary to sing songs. The main problem for the organization of the thematic evenings is the formation of a song list that should be played and sung by participants. After all, it is difficult to develop a list of songs that will sound throughout the evening taking into consideration all the factors that can affect the mood of its participants. This is because making a preliminary plan for the singing evening, it is impossible to guess what mood the participants will have. Since this problem is constantly present, it would be convenient to have a system that would help to form and select a list of songs according to the user needs. It is also a big plus that this system will be able to adapt to user behavior and change their offers. By collecting all these requirements, in the end, you will get an excellent system that responds to user feedback, takes them into account, and adapts its output results under it. The main goal of the work is to create an intelligent system following the user needs. The basic user needs were determined by his/her mood. Since the mood of the user of an intelligent system can constantly change, the system itself must also change its proposals in accordance with changes in mood. That is, the selection of songs will occur taking into account the mood of the user and the mood of the song. However, the user should always be able to choose a song with quite a different mood, which in turn should change the following tips for selecting songs. That is why this intelligent system will find an extremely wide audience of users.

The main task to be solved in the course of this work is to create an algorithm to determine the user mood, which should be based on the history of the songs, namely on the previously opened songs. It should be noted that the user may be in the song search phase and therefore it is necessary to consider the song opened by the user only after certain special conditions are met. The next task is to create an algorithm to determine the mood of the song. The mood of the song should be determined not only by the keywords but also by the context of the song, so another task is to clarify the context of the song. One more important step is to combine these algorithms for the most correct functioning of the intellectual system [8-12]. The object of research is real events in which there is a component of live singing. The main type of event is non-mainstream music evenings. In fact, during bard evenings, there is often a problem about which song to play and sing next. In this case, this intelligent system should help to solve this problem. The subject of the study is an algorithm that can be used to determine the emotional mood of a song. Another subject of research is an algorithm that will help to know the user mood at the moment and will help to choose and offer a particular song. However, the task is to choose the song according to your mood, sometimes it is very difficult. This intelligent system can automatically detect the user mood and suggest the next song. After investigating the current market for the electronic reference songs, it turned out that there is no solution to this problem yet. Therefore, it was decided to make an intelligent system that could determine the user mood, define the song mood and as a result, offer the user the next song. Lyrics and chord analysis were used to determine the song mood. This mood is characterized by a positive tone (cheerful, sublime) or negative (sad, depressed), as well as by different dynamics. It is known, to determine the mood of the lyrics you can use words and phrases that are usually occurred in respect to a certain type of mood. Words such as love, happiness, sun, mother are used for a cheerful mood. Words like

sadness, separation are used for a depressed mood. Besides, to determine the mood of the song, analysis of the chords that the song is played on was used. It is well known that funny, cheerful songs are performed on major chords. On the contrary, sad, melancholy songs are performed on minor chords. Since all the songs in the collection of songs that I have developed have chords, then determining the mood of songs with chords has a lot of potentials. Both methods are used to determine the mood of a song. In the basic implementation, a system of points was programmed that provides songs that can characterize its mood. To determine the user mood, an algorithm was used in which the user also has a rating scale, and by opening the next song, it corrects it. According to the user rating scale, the most similar song is selected according to its ratings and offered to the user.

2 Analytical Review of Literature Sources

A song is a verbal and musical work intended for singing [1-3]. The main indicators of a song as a genre of lyric-strophic structure, repetition of verses of the stanza, differentiation of chorus and refrain, expressive rhythmization, the musicality of sound, syntactic parallelism, simple syntactic structure. The oldest traditional variety of song lyrics uses different language means and conveys the best impressions. A folk song is distinguished as a genre of written poetry and as an independent vocal and musical work. The mood is a relatively stable mental state characterized by the presence of a common emotional background that determines the occurrence and course of various experiences and significantly affects human behavior. Mood differently affects the mental processes occurring in humans. In contrast to feelings that are always aimed at a specific object (present, future, past), mood, often caused by a specific reason, a specific event, is manifested in the features of the emotional response of a person to the impact of a character. The mood is characterized by a positive tone (cheerful, sublime) or negative (sad, depressed), as well as different dynamics. A relatively stable mood arises from satisfaction or dissatisfaction with a person's significant requests and aspirations. Among the factors that determine individual differences in people regarding the speed of mood change and other features, an important place goes to the characteristics of temperament. The opinion that the mood reigns over the person and is not subject to volitional control is not entirely true, because, with careful self-analysis, you can identify the factors that led to a certain mood. If the mood is expressed by a negative emotional background (bad mood), you can identify the factors that caused it and try to eliminate them. A person with a strong enough will can involuntarily influence their mood trying to ensure that it does not have a negative impact on the behavior. The ability to manage your mood is formed gradually and is an integral part of the training. Therefore, the mood is a total characteristic of a person’s emotional state [4].

The mood can be smooth (ethymic), excited (hyperthymic), lowered (hypothymic), anxious, and so on. The general tone of mood is largely determined by the course many mental processes, including thinking. Diffuseness, lack of clear conscious attachment to certain objects or processes, and sufficient stability allow us to consider mood as a separate indicator of temperament. Unlike situational emotions and affects, the mood is an emotional response not to the immediate consequences of

specific events, but to their significance for the subject in the context of general life plans, interests, and expectations. This mood, in turn, can influence emotional responses depending on what is happening, changing the direction of thought, perception (social perception), and behavior accordingly. Depending on the level of awareness of the reasons that caused a certain mood, it is perceived either as an indivisible general background (elated, depressed mood) or as a clearly defined state (boredom, sadness, longing, fear or admiration, joy, etc.). The ability to control the mood, find and study ways to consciously correct it (self-regulation) is an important task of education and self-education. Gratuitous mood swings can have a pathological origin due to such psychological properties as increased anxiety, instability, emotionality and others (character, accentuation, feeling). The mood is a general emotional state that initially colors a person activity for a certain time and characterizes his/her life force. There are positive moods that manifest themselves in cheerfulness, and negative moods that suppress, demobilize and cause passivity. The mood is a general emotional state that is definitely not directed at anything specific.

The reasons for the mood are very different: lack of preparation for action, fear of expected failure, painful conditions, good news, and so on.

Superstition occupies a special place among the causes of mood. Belief in omens, especially negative, causes passivity, fear, and upsets the mental activity of a person.

The measure of compliance with moods is individual. People, who have self-control, do not give in to the mood, do not get discouraged, even if there are reasons for this, but, on the contrary, struggle with difficulties. The faint-hearted quickly succumb to the mood. They need the support of the team.

3 System Analysis and Justification of the Problem

3.1 System Analysis of the Research Object and Subject Area

The objective of the work is to create an intelligent system that will identify and select the songs according to the user needs. In fact, an intelligent system that is crude will not interest the user. Therefore, it was decided to create an intelligent system based on mobile software, which will feature songs and chords [13-21].

This mobile software application will implement an intelligent system for selecting songs according to the user requests and mood. The intelligent system will be an algorithm for analyzing song lyrics. The algorithm must be able to get the context of the song and indicate its mood according to the context of the song. There are also several improvements to the song's intelligent mood analysis system. The most obvious thing is to participate in the analysis, in particular the analysis of the chords of a song, because in music you can hear that different melodies show the mood differently. For example, the vast majority of minor chords are responsible for a sad, depressed mood. However, the main chords often show an upbeat and solemn mood. A minor (secondary) triad is a triad consisting of a small third at the bottom and a large third at the top, between the extreme sounds of which an interval of a pure fifth is formed. The minor triad represents the tonic (a function of the main component) of the minor system. The major triad is a triad consisting of a large third at the bottom and a small third at the top, between the extreme sounds of which an

interval of a pure fifth is formed. A major triad is one of the main chords that includes sounds separated from the main tone by a major third and a pure fifth. There are also large major and minor major seventh chords.

Besides, this intelligent system will contain many other important features that will help the user in using this software product and the intelligent system as a whole. One of the important functions that should be implemented in this software product will be an automatic chord detection function. This functionality will be especially useful for those who often add their songs either from the Internet or from printed versions of the songwriter. Another useful feature of this intelligent system will be the ability to click a chord to see its tablature. Tablature is a form (diagram) of an instrumental record that uses letters or numbers instead of notes (or together with notes). In a typical tablature, the melody is shown on several horizontal lines (in the case of guitar tablature) corresponding to the guitar strings; the notes are indicated by fret numbers and arranged sequentially behind the fates. This functionality is extremely important for those who are just starting to play a musical instrument and do not fully know all the chords and finger positions. People who are just beginning to learn to play a musical instrument can also use this feature for educational purposes. Another equally important function is to transpose chords in a song. Chord transposition is the translation of the entire key of a song from one to another while preserving all the musical stances. This task is complex and at the same time important for solving the fact, that many songs are played using complex chords that most beginners can't play. This is why it would be good to have an automatic intelligent system that automatically changes the key. Today, most actions with chord transposition are performed manually, the chords are written somewhere separately, these chords are transferred to the desired key, and then the chords are written back according to the places where they were taken. Therefore, the implementation of this functionality will be very useful and will simplify the experience of using the app for many users.

The intelligent system will have many useful functions [22-27]. However, they are all aligned if they all work only if there is an Internet connection. Therefore, it will be necessary to develop the ability to work with this intelligent system in a mode when there is no Internet connection, namely in offline mode. The offline mode should not limit the functionality of both the smart system and the mobile software in general. Therefore, all calculations and intelligent calculations must be performed on the side of the intelligent system. The only place to connect to the Internet is a place to collect and transmit analytics of user actions. Analytics is necessary to determine how useful a particular functionality is for the user [28-32].

After a detailed analysis of the user actions, it will be possible to conclude about the appropriateness of the content of a particular function of the software product. The software product should be simple and easy to use. This means that the user needs to complete the smallest number of steps to execute the main actions. According to this, the more important the functionality, the fewer steps there should be to achieve it. To do this, you need to develop a mobile app, design a database, and fill it with data, explore this subject area, explore the information system, design and develop [33-41].

During the development process, all decisions made regarding the information system must be carefully checked. It is necessary to correctly analyze the user mood, as well as the mood of the song because a poorly developed at least one aspect can

lead to improper functioning of the entire system, which in turn will lead to negative feedback from users, and this will lead to their loss, which will eventually lead to the depreciation of the information system and mobile software in general [41-51]. In addition, this is a waste of time and money to develop this intelligent system and mobile app.

3.2 Conceptual Model

A conceptual model is a model representing a system that consists of several concepts that are used to better perceive, represent, and understand how the system works at different levels. This is also a set of concepts. The term conceptual model itself is used to define the models that arise from the process of generalization or conceptualization. Most often, conceptual models are streaming abstractions that exist in the real world. In most cases of use, as shown in Fig. 1, the actor (the app user) is shown, and the functionality is available to him. The use case diagram shows how the user can use our intelligent system. It is also shown that some use cases can be extended by other use cases or include other functions. Figure 2 shows the class diagram for the song preview window. The following functions are visible on the chart. The main display window is SongsActivity. It implements the SongView interface and contains an ISongPresenter. SongPresenter is a helper class that retrieves data from storage and displays it on the screen. This class implements the ISongPresenter interface. To design the behavior of the software product, a UML execution sequence diagram was developed, which is shown in Figure 3. This diagram shows the interaction between objects. The built diagram shows the user name, system (mobile app), and database. The diagram shows that the user interacts with the system using the application and the graphical user interface. The system, in turn, receives a request from the user and forms a request to the database. The database then processes the request from the system and returns it to the system. The system checks data from the database, formats it, and returns it to the user. Figure 4 shows the state diagram of the intelligent user mood recognition system. The state diagram starts when the app starts. After launching, the app works normally. When the user decides to open a song, the software switches to the song display state. At the same time, the intelligent system tries to determine the user mood based on the current song. Being able to show a song, the program can remain in the same state, opening the next song or the next one offered to the user. Alternatively, the program can switch to the song editing state. After that, as soon as the user has edited the song, the program performs mood analysis and determines the chords in the song.

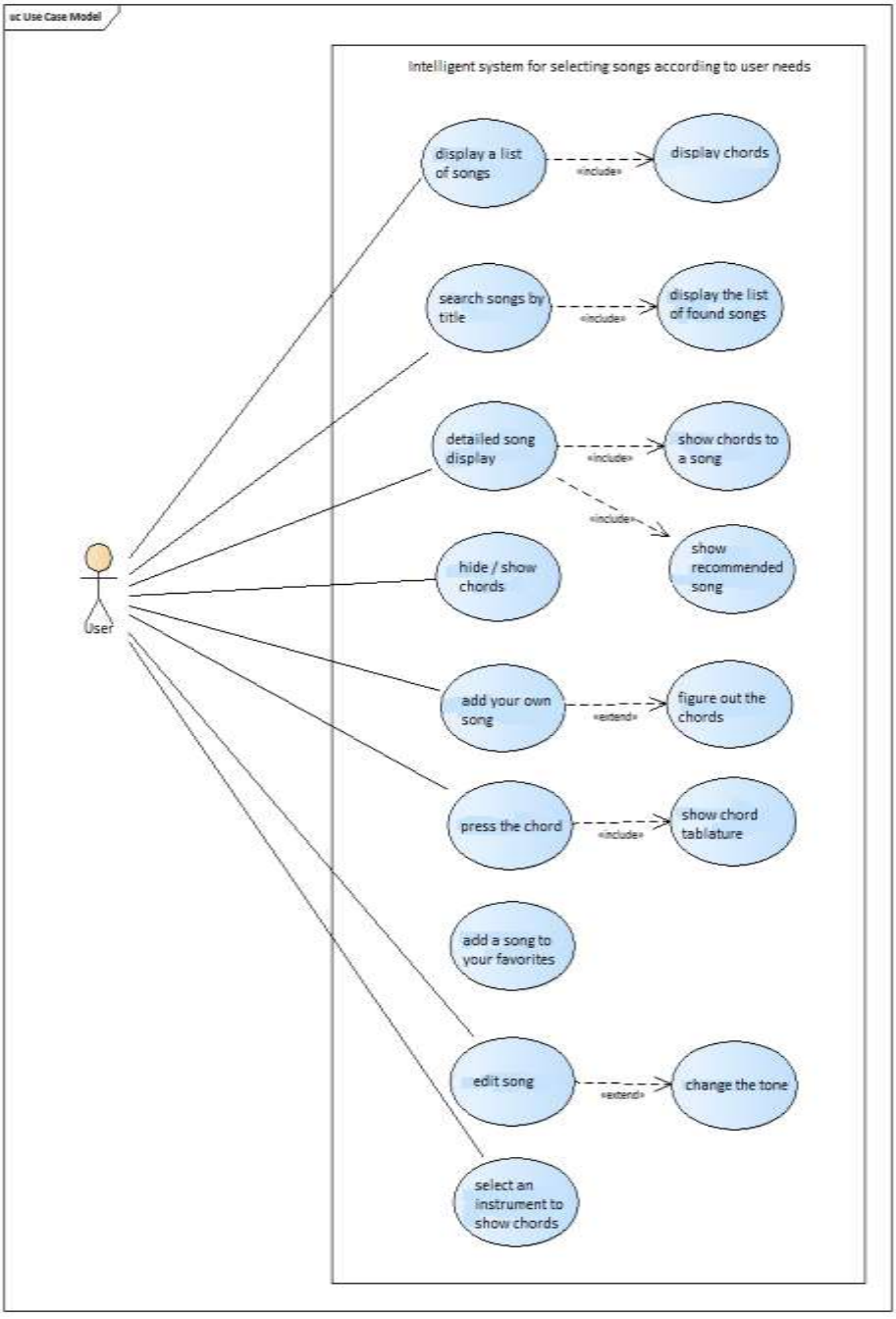


Fig. 1. Use case diagram

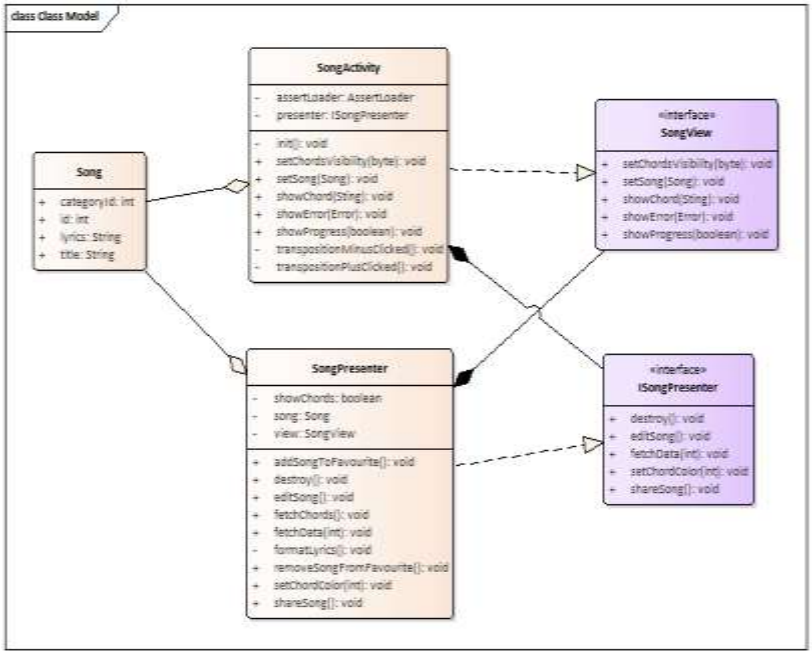


Fig. 2. Class diagram

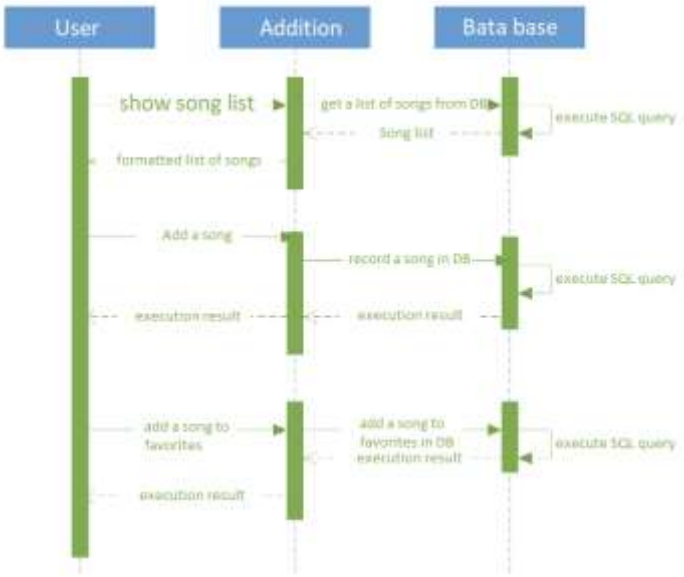


Fig. 3. Execution sequence diagram

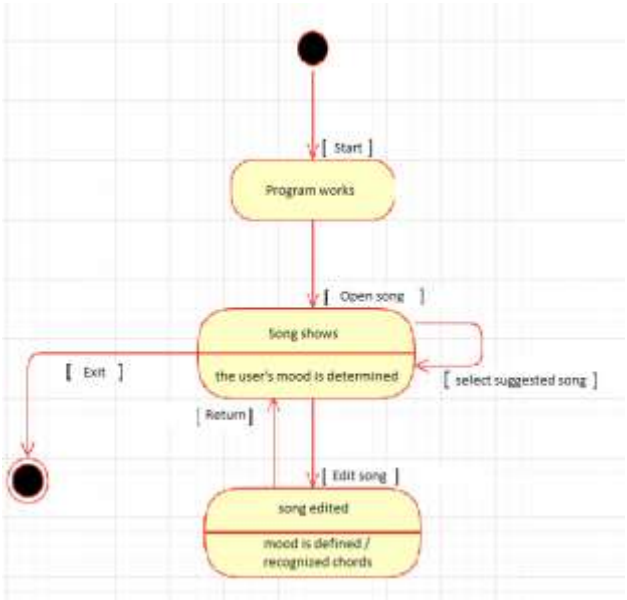


Fig. 4. State Diagram

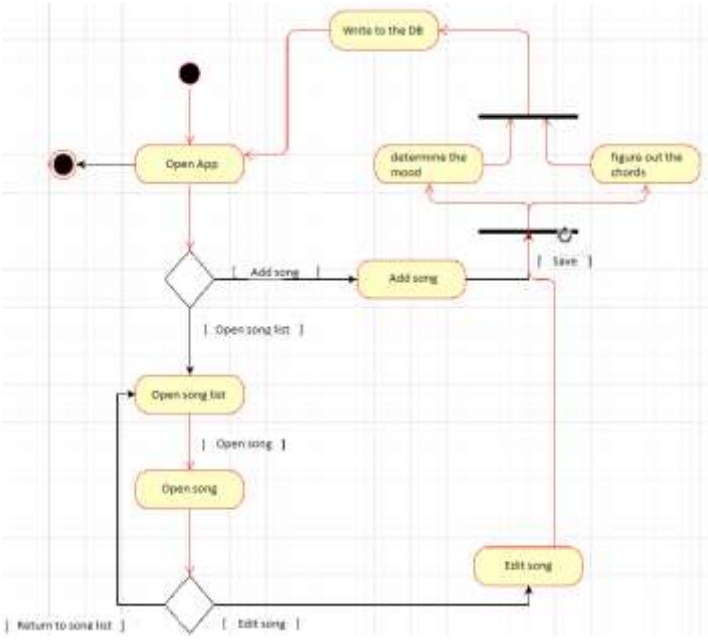


Fig. 5. Program activity diagram

The user can decide to exit the software product at any time by closing it.

Figure 5 shows the program activity diagram. It illustrates the activity of the app and each of its actions. Figure 6 shows the deployment diagram for the software product you are developing.

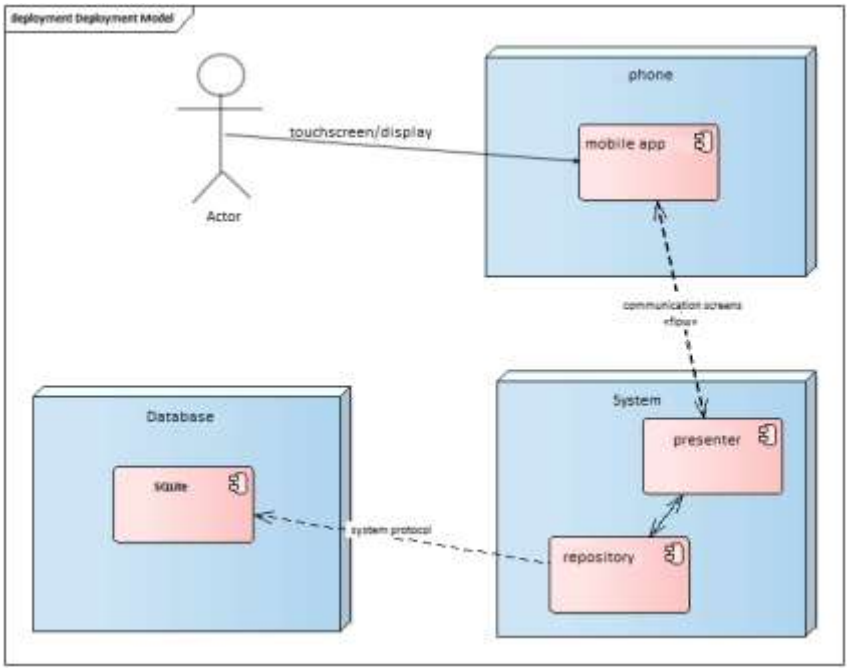


Fig. 6. Deployment diagram

This diagram shows exactly how the system is deployed. The user interacts with the system using a touch screen and display. The system processes input data from the user and sends it to the speaker. In turn, the speaker contacts the repository to request the necessary data. The storage, in turn, decides where it should get data. Since the database is currently the only data source, data will be extracted from the database accordingly. The request passes through the system protocol. The SQLite database will be taken as a database.

3.3 Statement and Justification of the Problem

For the success of the bard chants, it is necessary to properly plan the entire program of the event. The main component of the bard night, without which this evening is not complete, are the songs and exactly how they are sung. However, it often happens that the participants who came to the bard night do not know the words of the song. In addition, those who are new to the bard party may not even know a single song. To avoid such awkward situations, the organizers of bard nights often resort to providing with their theme songwriter with a set of songs that will be played at the bard night.

Then the organizers distribute these singers to all participants of the bard night. However, since the printing of songwriters is not free, organizers must either make a tab to participate in singing to cover the cost of printing of songwriters. However, such actions give rise to many problems. One of the most obvious problems is that a frequent participant in the singing evening will have a large number of songwriters, since each time they come to the bard evening, they will be given a printed songbook with songs. Therefore, after attending several such events, he will get many songbooks. In addition, a frequent participant of such evenings will probably know most of the songs, and they will not need this collection of songs. Besides, paying for participation in the bard night will reduce the number of participants, since the overwhelming number of participants are students. Every penny is important for a student. However, you can solve this problem quite simply by making the tab voluntary. That is, if a person came to bard night and does not have his/her songwriter, then she can buy it. However, after the previous problem has been solved, we are faced with the next one. People often forget things. Therefore, it is logical to assume that people might also forget to take songwriters on bard nights. It also happens that the decision to go to the bard evening is spontaneous, and the person will not have a songbook. Of course, you can buy a songbook at the bard party, but it is a waste of time and you do not want to buy it at home. Thanks to this, people who did not accept the songbook will not be able to sing along to everyone and feel uncomfortable.

Another problem is that often people who end up writing songs for the bard nights do not think about what will become their songbook for readers and musicians. Often compilers just copy songs from the Internet without thinking about whether the chords of the song are correct. Besides, after inserting a song into a songbook, the composer cannot format the song, align chords, or add the song to the content. All these factors will cause inconvenience when using sonics.

Often people who admire the singer see some errors or just realize that the chords presented in the author do not match the song, and they have a desire to edit the song. However, editing on the paper of the singer is almost impossible. Because to edit a song, you need to take a pen and start drawing on the paper of the song author. It is not convenient and not aesthetically pleasing. It would be much more beautiful if the user could fix the song and leave the same formatting as it was. In addition, the paper edition of the songwriter cannot automatically transfer chords. There are many songs with chords on the Internet at the same time. It just often happens that these chords are not correct or just not suitable for playing a particular instrument. In this case, it is useful to have a function for changing the song key. However, obviously, it is not possible to do it on a singer paper, but the simplest intelligent system can do it automatically without any human effort.

Last on the list, but not least, is the problem of using resources to create a paper songwriter. The point is that to create a printed version of a songwriter, you need to use paper, and the paper is made of wood. Because humanity cuts down a tree, it harms the environment. In order not to harm the environment and reduce deforestation, we need to reduce paper consumption. This is why an alternative paper songwriter will serve as a software counterpart.

3.4 Software as an Alternative to the Printed Panic

As described in paragraph 3.3, the songwriter has several disadvantages. So you need to think of software as an alternative to songwriters. The main types of software are mobile applications and web services. This is not surprising, because mobile apps and web services give people mobility and ease of use. The web service does not require any additional settings other than the Internet. However, in the modern world, people are connected to the Internet almost day and night. Mobile apps are convenient because they can work offline. After all, often bard nights can take place among the mountains or forests where there is no Internet connection, and then offline mode comes to the rescue. To date, there are not many analogue applications on the market that have their characteristics. We analyzed the most popular foreign and domestic analogues and highlighted their advantages and disadvantages.

1) Website "Ukrainian Song". The largest Ukrainian web service with songs from Ukraine. The purpose of the site "Ukrainian songs" is to collect a collection of the best Ukrainian songs from present to ancient times. Also information about their performers and authors. Most of the songs feature guitar chords. There are plans to place audio recordings of songs and notes to them.

Advantages:

- The largest database of Ukrainian songs and chords
- Search for songs
- Forum
- A large Association of people who constantly fill the site content and add new songs to it

Disadvantages:

- Only in the web version of the site

2) Mobile application "Near the fire-Songbook". The only application-songwriter with Ukrainian songs. There are paid and free versions of songwriters on the Market.

Advantages:

- Work in offline mode
- Chords to songs
- Ability to change text size
- Light and dark theme

Disadvantages:

- Old design
- Guitar chords only
- There is no possibility to edit the song
- No ability to add your songs

- No ability to search for songs
- No ability to make a list of favorite songs

3) Mobile application “Songs for guitar Rus”. A Russian developer wrote a mobile app. The app sends requests to the Internet service from which it downloads songs and saves them for offline display.

Advantages:

- A greater number of songs from the Internet
- Work offline
- Chord index
- Ability to create a list of favorite songs
- Ability to create playlists

Disadvantages:

- Contains ads
- Awkward search
- Paid version with additional functionality
- Small number of Ukrainian songs

4 Choice and Justification of Methods of Solving the Problem

When writing an application for the Android operating system, there are different approaches and architectural solutions. Each of them has its advantages and disadvantages. We analyzed several approaches, among which two were selected. We chose to use the clean architecture approach and the MVC (Model-View-Presenter) template for the view level. The main advantages of choosing these two architectures were ease of expansion and independence of one component from the other. Clean architecture is a software development philosophy that separates design elements at the ring level. The basic rule of clean architecture is that code dependencies can only come from the outer layers inside. The code on the inner layers may not know the functions on the outer layers. Variables, functions, and classes (any entities that exist in external layers cannot be mentioned at more internal levels. It is also recommended to format data between separate levels [13].

The clean architecture was created by Robert C. Martin and is advertised on his blog of uncle Bob. Like other software development principles, clean architecture attempts to provide a methodology for use in coding to facilitate the development of high-quality code that works better, is easier to change, update, and has fewer dependencies.

An important goal of clean architecture is to provide developers with a way to organize code so that it encapsulates business logic, but keeps it separate from the delivery mechanism. Visually, the levels of clean architecture are ordered into an indefinite number of rings. The outer levels of the rings are the lower-level mechanisms, while the inner, higher level contains the policy and entity.

Another important advantage of the network architecture is the ease of code testing since all components are independent of each other.

For the presentation level, the MVP architecture approach was implemented for convenience and flexibility.

The model-view-presenter (MVP) is derived from the model-view-controller (MVC) architectural template and is primarily used for creating user interfaces. [14]

In MVP, the host accepts the "average person" functionality. In MVP, the whole presentation logic is pushed forward.

MVP is a user interface architecture schema designed to facilitate automated block testing and improve problem separation in presentation logic:

A model is an interface that defines the data that will be displayed or otherwise processed in the user interface.

- A view is a passive interface that displays data (model) and directs lead-to-lead commands (events) to affect this data.
- The host affects the model and view. It extracts data from the storage (model) and formats it to display in the view.

Typically, a view implementation creates a specific object for the presentation by providing references to itself.

The main advantage of this approach is that the code is divided into logical units.

5 Database Design

The ERwin Data Modeler tool was selected to create a logical database model. Erwin Data Modeler is a computer program for data modeling. Originally developed by Logic Works, Erwin was acquired by a number of companies and then closed by the private equity firm Parallax Capital Partners, which acquired and merged it into a separate company, Erwin, Inc. Famularo. [15]

The software engine is based on the IDEF1X method, although it now supports diagrams related to the engineering information technology notation option, as well as dimensional modeling notation. A database for an intelligent system should not be complex and contain the following objects:

- Song. It contains information about the song, namely: ID, title, words with chords, song mood, and a link to the category number.
- Category. All songs will be assigned a specific category. At the beginning, there will be four categories: Patriotic songs, Interesting foreign songs, Near the fire and Something of your own. This entity will have the following fields: ID, name.
- Favorite. This object will store IDs of the app user's favorite songs. The entity will have the following fields: ID and the ID of a favorite song.
- History. This entity is necessary so that the user can quickly navigate between pre-sung songs. This entity will have the following attributes: ID and song ID.

Because no many-to-many bindings were used in the logical design of the database, the logical and physical schemas of the database correspond to each other. The diagram is shown in Fig. 7.

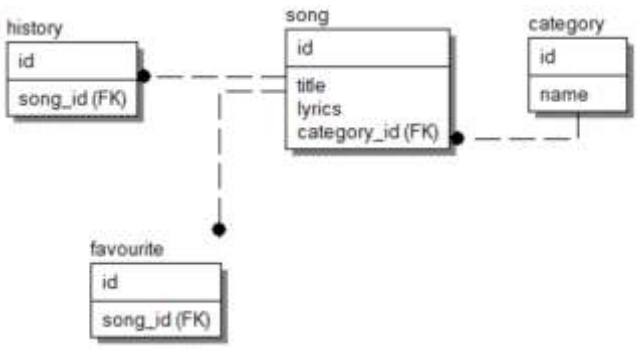


Fig. 7. Logical and physical database models

6 Creating Interface Prototypes

After the main functionality of the smart system was defined, we made an initial software map that demonstrated the structure of the future smart system. Its main frequent features are:

- Main window of the program
- Settings window
- A window with a search for songs and an image of all the songs
- Full lyrics with suggestions for the following songs
- A window with the ability to add and edit a song

The next step was to create the first prototypes for the Songbook mobile app. Figure 8a shows a prototype of the main program window. When the user opens the program, it goes to the main window of the program. The main window has buttons for quick access to all songs or a specific category of songs. Also in the main window of the program, there are transitions to the song-editing window, as well as to the application settings window. After the user has selected a certain category of songs or all the songs, it gets in the window with the list of songs. In this window, the user can search for songs by category or select one from the list. Besides, all songs are sorted alphabetically. To go to the current song, the user must click on the selected song. Then he gets to the window with the details of the song. The prototype of this window is shown in Fig. 8b. While on this page, the user sees the full text of the song, its name, and the chords marked and pressed. The song also offers one or more songs depending on the user mood. The Sketch software product was chosen to develop the user interface. Sketch is a vector graphics editor developed by the Dutch company Bohemian Coding. The Sketch was first released on September 7, 2010, for MacOS. It received the Apple Design Award in 2012. The main difference between Sketch and other vector graphics editors is that Sketch does not include print design features. The Sketch is only available on macOS. This problem is partially resolved by third-party tools and migration tools.

a) 

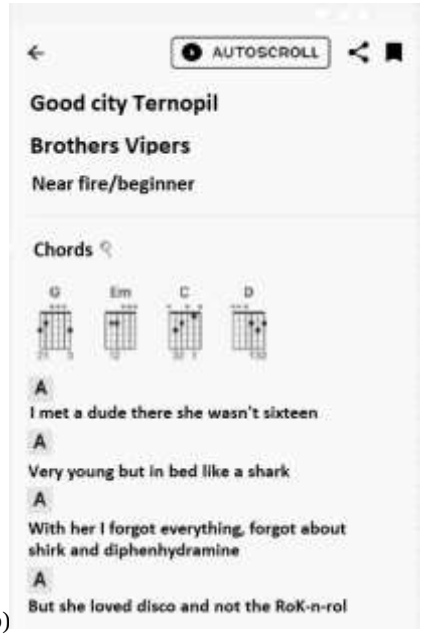
b) 

Fig. 8. Prototype of a) the main program window and b) the song preview window

7 Choice and Jjustification of Means to Solve the Problem

After developing the graphical user interface, all sketches were exported to the Zeplin software tool, since this software tool is convenient for the developer when developing and implementing all graphic components created by the designer [17].

The mobile app was decided to be developed for the Android OS operating system. The object-oriented Java language was chosen as the main programming language, as well as for the representation level of the statically typed Kotlin language. Because it was chosen to combine these two languages, the creation of a flexible and stable architecture was achieved simultaneously. The decision to develop a mobile app for the Android operating system was made for a number of reasons:

1) Android is considered one of the most popular mobile operating systems, with about two billion active users of this system.
2) Android SDK contains open source code that can be downloaded and used for free.
3) A huge number of free libraries on the Internet.
4) Built-in SQLite database at the core level, which allows you to query the database at high speed.
5) Android is an operating system created by Google based on the Linux kernel.

The development environment was chosen as Android Studio, built on the IDEA developed by JetBrains. This environment is also the official development environment for Android OS.

8 Task Implementation Description

To meet the main functional requirements of the product being developed, you need to design and develop a database. For the development of SQLite database models, a convenient and free environment for the MacOS operating system, SQLiteStudio, was chosen. This software product was chosen for several reasons:

- The software is available for the MacOS operating system because this operating system has developed an information system to generate a list of songs according to the user needs.
- The software tool is free of charge.
- The software contains all the necessary functions for creating and filling in the database.
- The software tool is easy to use and has an intuitive user interface.

Standard functions and methods of the Android operating system were used to transfer the created SQLite database file to the developed information system. Their main advantages are simplicity and speed of execution, as well as the ease of maintenance of the software product in the future. To provide access from the database information system, the library uses object-relational mapping, which is produced by Google, which developed the Android operating system, which is called Room. Room has the following main advantages:

- A small amount of writing so-called standard code, which is repeated in many places, to implement simple queries of the information system to the database.
- Checking and error checking of SQL queries at the compilation stage, which guarantees that there is no risk of errors during program execution, which leads to the shutdown of the information system.

9 The Installation of the Application Architecture

9.1 Development of Interactors and Repositories

After successfully creating database models and then filling them with data, you need to think about ways to communicate, transfer data from the database to the user view. Therefore, to implement this information system, we chose to use a “Clean architecture”. First, we need to create an interactor. The interactor contains all the business logic of the information system. The interactor should not have any dependencies on the environment in which it runs, or the resources to which it is accessible. The advantages of such a “clean” interactor are that it is extremely easy to test. It is enough to write the usual unit tests. To run these tests, the user only needs to

run them on a Java machine. Android environment is not required. Another advantage is that if the requirements of the information system change, it is quite easy to change the database implementation.

```
public class SongInteractor {
    private final SongRepository songRepository;
    private final FavouriteRepository favouriteRepository;
    private final HistoryRepository historyRepository;
    private Song deletion;
    public SongInteractor(SongRepository songRepository,
                          FavouriteRepository favouriteRepository,
                          final HistoryRepository historyRepository) {
        this.songRepository = songRepository;
        this.favouriteRepository = favouriteRepository;
        this.historyRepository = historyRepository;
    }

    public Single<List<Song>> getAllByCategory(
                         @Category.CategoryId int categoryId) {
        switch (categoryId) {
            case Category.FAVOURITE_ID:
                return favouriteRepository.getAll();
            case Category.LAST_ID:
                return historyRepository.getLastSongs();
            case Category.ALL_ID:
                return songRepository.getAll();
            case Category.ABROAD_ID:
            case Category.BONFIRE_ID:
            case Category.PATRIOTIC_ID:
            case Category.USERS_ID:
            default:
                return songRepository.getAllByCategory(categoryId);
        }
    }
    public Completable insertOrUpdate(Song song) {
        if (TextUtils.isEmpty(song.getTitle())) {
            return Completable.error(
                          new IllegalArgumentException(
                          "Title must be not empty"));
        }
        if (TextUtils.isEmpty(song.getLyrics())) {
            return Completable.error(
                          new IllegalArgumentException(
                          "Lyrics must be not empty"));
        }
        return songRepository.insertOrUpdate(song);
    }
}
```

The interactor uses storage to communicate with the database. The repository itself is an interface that has all the necessary methods for accessing the database. Then a special repository interface was implemented for accessing the database using ORM Room. The Room uses the data access DAO object to access data from the database. The DAO is a simple interface with methods and each method must be commented on. The annotation parameter is a query to the database. What the query returns must also return the method that this annotation was written to. You can also pass the function parameters to the query using the ":" operator. An example of the DAO interface for adding a list of songs is shown below:

```
@Dao
Public interface SongDao {
    @Query("SELECT * FROM song ORDER BY song.title")
    Single<List<SongEntity>> getAll();
    @Query("SELECT * FROM song WHERE
            song.category_id = :categoryId ORDER BY song.title")
    Single<List<SongEntity>> getAllByCategory(int categoryId);
  ***
}
```

9.2 Creating Database Queries

Since the main source of data in the software product being developed is a database, it is logical that you need to register requests to it well. Queries should be simple and comprehensive at the same time. To access the list of all songs, you need to go to the table with songs, which is called “songs” and run the following query:

```
SELECT * FROM song ORDER BY song.title
```

This query returns all the songs in the database, sorted by song name. You can also write a query to select only those songs that fall under a certain category. To do this, the WHERE operator was added with a comparison of the category ID:

```
SELECT * FROM song WHERE song.category_id = :
categoryId ORDER BY song.title
```

Another feature of this software product is to search for songs by certain keywords. The search must be performed both by the song name and by the song text. To do this, the LIKE keyword was used with a combination of parameters passed to the function:

```
SELECT * FROM song WHERE
(song.title LIKE: query OR song.lyrics LIKE: query)
AND song.category_id = :categoryId ORDER BY song.title
```

The result of this query is also returned sorted by song title in ascending order. You also need to add songs to your existing song list. Cui functionality is implemented using the INSERT query. Deleting is performed using the DELETE operator.

9.3 Creating a Presentation

After the database is developed and populated, and interactors and repositories are developed and encoded, the user interface development should begin. Components such as Activity and Fragment are responsible for creating the user interface in the Android operating system. Use Activity was selected. The XML markup code is located in a separate file with *domain.xml and binds to the Activity using the setContentView method. As planned, the user interface should be as simple and intuitive as possible. To act, the user must make a minimum number of clicks.
To place graphical components in an Activity, you can use program code or, more often, XML markup code, the code shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <com.google.android.material.appbar.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </com.google.android.material.appbar.AppBarLayout>
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <com.roman.kubik.songer.presentation.view.MainItemView
                android:id="@+id/lastCategory"
                android:layout_width="match_parent"
                android:layout_height="@dimen/view_size_72"
                app:image="@drawable/ic_history"
                app:title="@string/ttl_last_played"
                app:description="@string/dsc_last_played"/>
            <View style="@style/AppTheme.Separator" />
               </LinearLayout>
    </ScrollView>
</LinearLayout>
```

The main colors of the GUI are gray, white, and olive. These are the three colors that form the main color palette following the design recommendations from Google Material Design Guidelines. All colors are placed in a separate file called colors.xml. This file is located in the resources directory:

```
<resources>
    <color name="colorPrimary">#f8f8f8</color>
    <color name="colorPrimaryDark">#f4f4f4</color>
    <color name="colorAccent">@color/olive_green</color>
    <color name="colorBackground">#f8f8f8</color>
    <color name="colorOnPrimary">#d9d9d9</color>
    <color name="colorTextPrimary">@android:color/black</color>
    <color name="colorTextSecondary">#777777</color>
    <color name="hot_pink">#e8198b</color>
    <color name="olive_green">#417505</color>
    <color name="bright_green">#7ed321</color>
    <color name="aqua_marine">#50e3c2</color>
    <color name="darkish_grey">#9b9b9b</color>
    <color name="just_purple">#bd10e0</color>
    <color name="book_marked_category">#777777</color>
    <color name="my_songs">#FF9E30</color>
    <color name="tutorial_background">#cf000000</color>
    <color name="tutorial_on_background">#d9d9d9</color>
</resources>
```

As designed during the design stage, the main window of the program has quick access to the main features of the program, such as all songs, the latest favorite songs performed, and songs by category. Also in this window, you can go to the settings menu, where the user of this information system can choose whether he/she wants to show chords or not, choose a musical instrument on which it will be shown chords (Fig. 9a).

a) b) c)

Fig. 9. a) Main application window; b) All songs; c) The song details

If the user selects a specific category, a new program window opens, where they will see a list of songs from the selected category. The song list is displayed as the song title and the first four lines of the song. When loading a song, the software displays a loading indicator for the user (Fig. 9b). After clicking on a song, the user is redirected to the song details. Here the user sees the entire song, as well as the chords to it. Here he/she can change the key of the song and choose whether he wants to see the chords or not. Also at the end of the list is a list of recommended songs for the user, according to their needs (Fig. 9c). This selection of songs is handled by the corresponding information system, which will be described below.

On the top panel, the user sees a button for editing the song. After clicking on it, the user is redirected to the song-editing window.

10 Writing an Algorithm to Recognize Chords in Text

Since determining the mood of a song consists of two components: the keyword; chords. It is logical to come up with and implement an algorithm that will determine the chords among the lyrics. This task is reduced only to writing an algorithm for determining the location of chords among the text, and not to the actual analysis of the melody and automatic selection of chords. Accordingly, it was assumed that all the lyrics that would directly relate to the software product, as well as all the songs

that the user would add themselves, would already contain chords. From this assumption, it follows that the algorithm is reduced to a simpler and actually analyzing the text for special characters that represent chords.

After successfully searching for a chord in a song, you must mark it with special characters. For convenience, the tags will be called in the future. This action must be performed to simplify the chord recognition algorithm in the future.

This algorithm is implemented using the principle of regular expressions and their search in the text. The algorithm for first searching for chords and marking them in the text is presented below:

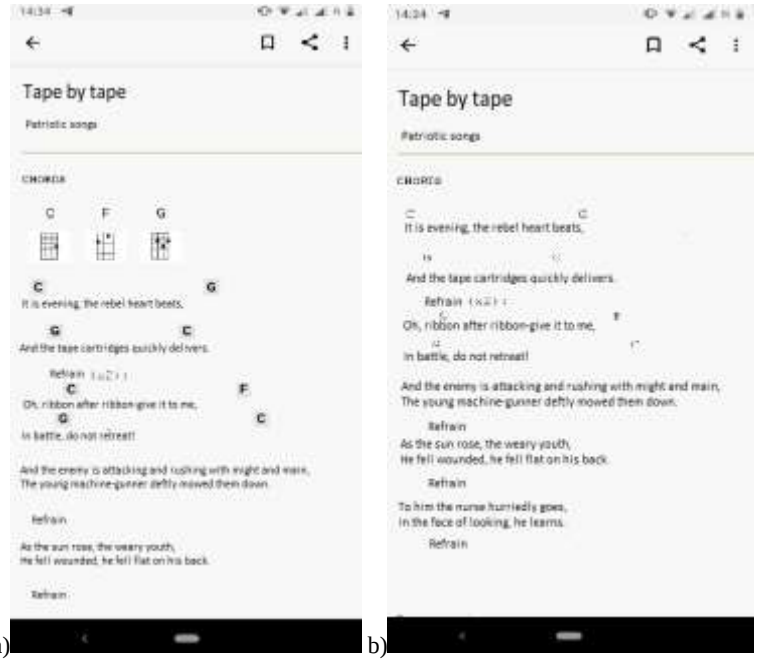
```
public CharSequence format(String text) {
        String result = text;
        for (String chord : CHORDS_GROUP) {
            for (String minor : MINOR_GROUP) {
                for (String sept : SEPT_GROUP) {
                    for (String accidental : ACCIDENTAL_GROUP) {
                        String fullChord =
                        chord.concat(minor).
                        concat(sept).concat(accidental);
                        result = findAndReplace(result, fullChord);
                    }
                }
            }
        }

        return result;
    }
```

The algorithm for recognizing chords in the text is implemented using four nested loops. The first cycle goes through all the known chords and finds them in the text. The next loop determines whether this chord is minor or major. Major chords do not have a prefix to the chord, but a minor has the prefix “m”. The third cycle determines whether it is a simple chord or a seventh chord. The seventh chord is marked with the special symbol “7”. The last loop checks the chord for changes. Changes are signs that increase or decrease the key of a chord. Within a letter, they are represented by the characters “#” for raising the key and “b” for lowering the key, respectively. The result of the algorithm is a highlighted song text, in which all chords are highlighted in a bold and gray background color. Figure 10 shows the result of the chord selection algorithm.

11 Analysis of the Results

To check the correctness of the developed software, you need to analyze the results. Methods such as validation and verification will be used to analyze the results. The essence of verification is to check whether the system is created as intended. The essence of validation is to check the correctness of the recorded algorithms, the correctness of the program operation under extreme input data. A set of test cases was developed for testing. The test cases were divided into two separate groups, each of which will have its narrow specialization. The first group is a group to verify the performance of the algorithm (validation code). The second group is a group for checking the visual representation of data to the end user, called GUI checking.



a) b)

Fig. 10. a) Song before processing by the algorithm. б) Song after processing by the algorithm

11.1 Unit Testing

To test the individual functions of the program code and independent parts of the code, we decided to use a Java code-testing environment called JUnit. JUnit includes a set of useful functions that help with unit testing. The main functions of JUnit are functions that check for equality of two values. Table 1 shows how to use unit testing:

Table 1. Use cases of unit testing

Use cases	Test cases	Test data
Removing chords from text	4	22
Chord recognition	5	25
Search songs by title	2	4
Edit song	2	2
Adding a new song	2	2
Getting a list of songs	1	6
Offer songs to the user	5	20
Total	21	81

No errors were detected during unit testing, which means that the algorithms are working correctly.

11.2 Testing the Graphical User Interface

The main purpose of testing the graphical user interface is to check whether the chords of the song, the actual title, and lyrics of the song are displayed correctly, as well as the correct navigation in the mobile app. To test all these graphical components, we created many test cases to test the program graphical user interface, which can be viewed in table 2 below.

Table 2. Use cases for interface testing

Use case	Test data
Checking the main program window	5
Check the window with a list of all songs	10
Checking the song details window	2
Check the window with a list of all songs by category	10
Check the add your own song window	2
Check the window showing the selected chord	3
Check the window showing the selected song	4
Checking the window with the suggested song	3
Total	39

During testing, the GUI has not discovered any deviation from the design.

11.3 Security Testing

Also, a separate point is to check the security of the software product because the user safety should be in the first place. Since the software product does not have access to the Internet, this means that it does not have potential threats to access data.

11.4 Verification of the Results

Verification of the obtained results is aimed at determining whether the software product is the product that was planned. Since the main task that was supposed to solve the intelligent system we developed, is to select songs according to the user mood, we can safely say that this software product completely solves this problem. Its main functions are also displaying a list of songs, displaying a list of songs by category, displaying a single song, editing a song, adding a new song, recognizing chords among the text, displaying a chord when clicked. The implementation of the mobile app shows that all the functionality specified above has been implemented.

Besides, according to the unit testing and testing of the graphical user interface, it can be concluded that this information system works.

12 Conclusion

In the course of the work, an intelligent system for selecting songs according to the user needs was designed and developed. The mood was chosen as the user main need because depending on the mood, people usually choose a specific musical composition to sing.

The qualification results are as follows:

- An analysis of the subject area in which the intelligent system should be developed. The main criterion for determining the user needs is defined. The user main need was determined by the mood.
- Different approaches to analyzing the mood of the song were analyzed. The main one is the analysis of the song keywords, the second one is the analysis of chords in the song. It has been investigated that minor chords tend to reflect a sad, depressed mood. But major chords are most often responsible for a cheerful mood.
- Various intelligent analogue systems were analyzed. From the analysis, it became clear that each of them has its advantages and disadvantages. Based on the results, it was decided that the developed intelligent system will combine its advantages, will not have its disadvantages, and will also have, in fact, an intelligent system for selecting songs according to the user needs.
- Knowledge of the subject area was systematized, after which it was decided to model the behavior of the system, as well as design the implementation of the software. Several UML diagrams were created for this purpose, including use case diagram, class diagram, sequence diagram, state diagram, action diagram, and deployment diagram.
- Developed prototypes of a graphical user interface that represents the main Window for user interaction. The graphical interface is designed in the Material Design style.
- Clean architecture was chosen as the main architecture of the intelligent system. The MPV template was selected for the presentation level.
- The intelligent system was implemented under the Android operating system. Android was chosen because it is one of the most popular mobile operating systems with open source code and a huge number of open libraries.
- To verify and validate the obtained results, the software product was tested to see if it was performing the task and if it had any system errors. The results of the portfolio and verification showed that the system behaves as planned.

The result is a fully functioning information system for selecting songs according to the user mood.

References

1. Vysotska, V., Lytvyn, V., Burov, Y., Gozhyj, A., Makara, S.: The consolidated information web-resource about pharmacy networks in city," CEUR Workshop Proceedings, Vol-2255, 239-255. (2018)
2. Korobchinsky, M., Vysotska, V., Chyrun, L., Chyrun, L.: Peculiarities of Content Forming and Analysis in Internet Newspaper Covering Music News, In: Computer Science and Information Technologies, Proc. of the Int. Conf. CSIT, 52-57 (2017)
3. Naum, O., Chyrun, L., Kanishcheva, O., Vysotska, V.: Intellectual System Design for Content Formation. In: Computer Science and Information Technologies, Proc. of the Int. Conf. CSIT, 131-138 (2017)
4. Kanishcheva, O., Vysotska, V., Chyrun, L., Gozhyj, A.: Method of Integration and Content Management of the Information Resources Network. In: Advances in Intelligent Systems and Computing, 689, Springer, 204-216 (2018)
5. Lytvyn, V., Vysotska, V.: Designing architecture of electronic content commerce system. In: Computer Science and Information Technologies, Proc. of the X-th Int. Conf. CSIT'2015, 115-119 (2015)
6. Vysotska, V.: Linguistic Analysis of Textual Commercial Content for Information Resources Processing. In: Modern Problems of Radio Engineering, Telecommunications and Computer Science, TCSET'2016, 709–713 (2016)
7. Rusyn, B., Vysotska, V., Pohreliuk, L.: Model and architecture for virtual library information system. In: Computer Sciences and Information Technologies, 37-41. (2018)
8. Rusyn, B., Lytvyn, V., Vysotska, V., Emmerich, M., Pohreliuk, L.: The Virtual Library System Design and Development. In: Advances in Intelligent Systems and Computing, 871, 328-349. (2019)
9. Shakhovska, N., Vysotska, V., Chyrun, L.: Features of E-Learning Realization Using Virtual Research Laboratory. In: Computer Science and Information Technologies, Proc. of the XI–th Int. Conf. CSIT'2016, 143–148 (2016)
10. Shakhovska, N., Vysotska V., Chyrun, L. Intelligent Systems Design of Distance Learning Realization for Modern Youth Promotion and Involvement in Independent Scientific Researches. In: Advances in Intelligent Systems and Computing 512. Springer International Publishing AG, 175–198 (2017)
11. Su, J., Vysotska, V., Sachenko, A., Lytvyn, V., Burov, Y.: Information resources processing using linguistic analysis of textual content. In: Intelligent Data Acquisition and Advanced Computing Systems Technology and Applications, Romania, 573-578, (2017)
12. Rzheuskyi, A., Gozhyj, A., Stefanchuk, A., Oborska, O., Chyrun, L., Lozynska, O., Mykich, K., Basyuk, T.: Development of Mobile Application for Choreographic Productions Creation and Visualization. In: CEUR Workshop Proceedings, 2386, 340-358. (2019)
13. Lytvyn, V., Vysotska, V., Rzheuskyi, A.: Technology for the Psychological Portraits Formation of Social Networks Users for the IT Specialists Recruitment Based on Big Five, NLP and Big Data Analysis. In: CEUR Workshop Proceedings, Vol-2392,147-171. (2019)
14. Vysotska, V., Chyrun, L., Chyrun, L.: Information Technology of Processing Information Resources in Electronic Content Commerce Systems. In: Computer Science and Information Technologies, CSIT'2016, 212-222 (2016)
15. Lytvyn, V., Vysotska, V., Chyrun, L., Chyrun, L.: Distance Learning Method for Modern Youth Promotion and Involvement in Independent Scientific Researches. In: Proc. of the IEEE First Int. Conf. on Data Stream Mining & Processing (DSMP), 269-274 (2016)

16. Vysotska, V., Rishnyak, I., Chyrun L.: Analysis and evaluation of risks in electronic commerce, CAD Systems in Microelectronics, 9th International Conference, 332-333 (2007)
17. Vysotska, V., Chyrun, L.: Analysis features of information resources processing. In: Computer Science and Information Technologies, Proc. of the Int. Conf. CSIT, 124-128 (2015)
18. Vysotska, V., Chyrun, L., Chyrun, L.: The Commercial Content Digest Formation and Distributional Process. In: Computer Science and Information Technologies, Proc. of the XI–th Int. Conf. CSIT'2016, 186-189 (2016)
19. Lytvyn, V., Vysotska, V., Pukach, P., Bobyk, I., Pakholok, B.: A method for constructing recruitment rules based on the analysis of a specialist's competences. In: Eastern-European Journal of Enterprise Technologies, 6/2(84), 4-14. (2016)
20. Chyrun, L., Kis, I., Vysotska, V., Chyrun, L.: Content monitoring method for cut formation of person psychological state in social scoring. In: Int. Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 106-112. (2018)
21. Chyrun, L., Vysotska, V., Kis, I., Chyrun, L.: Content Analysis Method for Cut Formation of Human Psychological State. In: International Conference on Data Stream Mining and Processing, 139-144. (2018)
22. Lytvyn, V., Vysotska, V., Burov, Y., Veres, O., Rishnyak, I.: The Contextual Search Method Based on Domain Thesaurus. In: Advances in Intelligent Systems and Computing, 689, 310-319 (2018)
23. Lytvyn, V., Vysotska, V., Veres, O., Rishnyak, I., Rishnyak, H.: Classification methods of text documents using ontology based approach, Advances in Intelligent Systems and Computing, 512, 229-240 (2017)
24. Su, J., Sachenko, A., Lytvyn, V., Vysotska, V., Dosyn, D.: Model of Touristic Information Resources Integration According to User Needs. In: Computer Sciences and Information Technologies, 113-116. (2018)
25. Lytvyn, V., Vysotska, V., Dosyn, D., Burov, Y.: Method for ontology content and structure optimization, provided by a weighted conceptual graph. In: Webology, 15(2), 66-85. (2018)
26. Gozhyj, A., Kalinina, I., Vysotska, V., Gozhyj, V.: The method of web-resources management under conditions of uncertainty based on fuzzy logic. In: Int. Scientific and Technical Conference on Computer Sciences and Information Technologies, 343-346. (2018)
27. Gozhyj, A., Vysotska, V., Yevseyeva, I., Kalinina, I., Gozhyj, V.: Web Resources Management Method Based on Intelligent Technologies. In: Advances in Intelligent Systems and Computing, 871, 206-221. (2019)
28. Lytvyn, V., Vysotska, V., Dosyn, D., Lozynska, O., Oborska, O.: Methods of Building Intelligent Decision Support Systems Based on Adaptive Ontology. In: International Conference on Data Stream Mining and Processing, DSMP, 145-150. (2018)
29. Burov, Y., Vysotska, V., Kravets, P.: Ontological approach to plot analysis and modeling," CEUR Workshop Proceedings, Vol-2362, 22-31. (2019)
30. Lytvyn, V., Vysotska, V., Veres, O., Rishnyak, I., Rishnyak, H.: The Risk Management Modelling in Multi Project Environment. In: International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT, 32-35. (2017)
31. Lytvyn, V., Vysotska, V., Pukach, P., Vovk, M., Ugryn, D.: Method of functioning of intelligent agents, designed to solve action planning problems based on ontological approach," Eastern-European Journal of Enterprise Technologies, 3/2(87), 11-17. (2017)

32. Lytvyn, V., Vysotska, V., Demchuk, A., Demkiv, I., Ukhanska, O., Hladun, V., Kovalchuk, R., Petruchenko, O., Dzyubyk, L., Sokulska, N.: Design of the architecture of an intelligent system for distributing commercial content in the internet space based on SEO-technologies, neural networks, and Machine Learning. In: Eastern-European Journal of Enterprise Technologies, 2(2-98), 15-34. (2019)
33. Vasyl Lytvyn, Victoria Vysotska, Dmytro Dosyn, Roman Holoschuk, Zoriana Rybchak: Application of Sentence Parsing for Determining Keywords in Ukrainian Texts. In: Computer Science and Information Technologies, CSIT, 326-331. (2017).
34. Vysotska, V., Hasko, R., Kuchkovskiy, V.: Process analysis in electronic content commerce system. In: International Conference on Computer Sciences and Information Technologies, CSIT, 120-123. (2015)
35. Vysotska, V., Fernandes, V.B., Emmerich, M.: Web content support method in electronic business systems", CEUR Workshop Proceedings, Vol-2136, 20-41. (2018)
36. Lytvyn, V., Sharonova, N., Hamon, T., Vysotska, V., Grabar, N., Kowalska-Styczen, A.: . In: Computational linguistics and intelligent systems. In: CEUR Workshop Proceedings, Vol-2136. (2018)
37. Lytvyn, V., Vysotska, V., Rusyn, B., Pohreliuk, L., Berezin, P., Naum, O.: Textual Content Categorizing Technology Development Based on Ontology. In: CEUR Workshop Proceedings, Vol-2386, 234-254. (2019)
38. Vysotska, V., Lytvyn, V., Burov, Y., Berezin, P., Emmerich, M., Basto Fernandes, V.: Development of Information System for Textual Content Categorizing Based on Ontology. In: CEUR Workshop Proceedings, Vol-2362, 53-70. (2019)
39. Leoshchenko, S., Oliinyk, A., Skrupsky, S., Subbotin, S., Zaiko, T.: Parallel Method of Neural Network Synthesis Based on a Modified Genetic Algorithm Application. In: CEUR Workshop Proceedings, Vol-2386, 11-23. (2019)
40. Antonyuk, N., Medykovskyy, M., Chyrun, L., Dverii, M., Oborska, O., Krylyshyn, M., Vysotsky, A., Tsiura, N., Naum, O.: Online Tourism System Development for Searching and Planning Trips with User's Requirements. In: Advances in Intelligent Systems and Computing IV, Springer Nature Switzerland AG 2020, 1080, 831-863. (2020)
41. Rzheuskyi, A., Kutyuk, O., Voloshyn, O., Kowalska-Styczen, A., Voloshyn, V., Chyrun, L., Chyrun, S., Peleshko, D., Rak, T.: The Intellectual System Development of Distant Competencies Analyzing for IT Recruitment. In: Advances in Intelligent Systems and Computing IV, Springer, Cham, 1080, 696-720. (2020)
42. Rusyn, B., Pohreliuk, L., Rzheuskyi, A., Kubik, R., Ryshkovets Y., Chyrun, L., Chyrun, S., Vysotskyi, A., Fernandes, V. B.: The Mobile Application Development Based on Online Music Library for Socializing in the World of Bard Songs and Scouts' Bonfires. In: Advances in Intelligent Systems and Computing IV, Springer, 1080, 734-756. (2020)
43. Antonyuk N., Chyrun L., Andrunyk V., Vasevych A., Chyrun S., Gozhyj A., Kalinina I., Borzov Y.: Medical News Aggregation and Ranking of Taking into Account the User Needs. In: CEUR Workshop Proceedings, Vol-2362, 369-382. (2019)
44. Chyrun, L., Chyrun, L., Kis, Y., Rybak, L.: Automated Information System for Connection to the Access Point with Encryption WPA2 Enterprise. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 389-404. (2020)
45. Kis, Y., Chyrun, L., Tsymbaliak, T., Chyrun, L.: Development of System for Managers Relationship Management with Customers. In: Lecture Notes in Computational Intelligence and Decision Making, 1020, 405-421. (2020)
46. Chyrun, L., Kowalska-Styczen, A., Burov, Y., Berko, A., Vasevych, A., Pelekh, I., Ryshkovets, Y.: Heterogeneous Data with Agreed Content Aggregation System Development. In: CEUR Workshop Proceedings, Vol-2386, 35-54. (2019)

47. Chyrun, L., Burov, Y., Rusyn, B., Pohreliuk, L., Oleshek, O., Gozhyj, A., Bobyk, I.: Web Resource Changes Monitoring System Development. In: CEUR Workshop Proceedings, Vol-2386, 255-273. (2019)
48. Gozhyj, A., Chyrun, L., Kowalska-Styczen, A., Lozynska, O.: Uniform Method of Operative Content Management in Web Systems. In: CEUR Workshop Proceedings, Vol-2136, 62-77. (2018)
49. Chyrun, L., Gozhyj, A., Yevseyeva, I., Dosyn, D., Tyhonov, V., Zakharchuk, M.: Web Content Monitoring System Development. In: CEUR Workshop Proceedings, Vol-2362, 126-142. (2019)
50. Veres, O., Rusyn, B., Sachenko, A., Rishnyak, I.: Choosing the Method of Finding Similar Images in the Reverse Search System. In: CEUR Workshop Proceedings, Vol-2136, 99-107. (2018)
51. Basyuk, T., Vasyliuk, A., Lytvyn, V.: Mathematical Model of Semantic Search and Search Optimization. In: CEUR Workshop Proceedings, Vol-2362, 96-105. (2019)