음원 프로그램 명세서

2019.12.04

2018102694
컴퓨터공학과 서예진

목차

1. 프로젝트 명

음원 관리 프로그램

2. 서론

(1) 개요

본 문서는 음원관리 프로그램 서비스에 대한 소프트웨어 요구사항 명세서이다. 본 서비스를 위한 요구사항을 정리, 분석하고 기재된 내용을 바탕으로 시스템을 설계 및 구현한다.

본 문서는 음원관리 프로그램을 이용할 사용자를 주 독자로 한다. 프로그램은 본 명세서에 따라서 음원 정보를 효율적으로 관리하여 서비스를 제공 할 수 있는 정보관리 시스템을 설계하고 구현한다. 추후 본 시스템을 상품으로 개발할 경우, 이와 관련된 모든 업체의 직원들이 추가적인 독자가 될 수 있다.

(2) 목적

본 명세서에서 작성한 음원관리 프로그램은, Sorted Linked list 의 template class 를 구현하여 여러 TYPE 의 class 를 유동적으로 활용하는 것을 목적으로 한다. 또한 음원 정보 관리 시스템을 위한, 세부적인 기술들의 각각에 맞는 효율적인 자료구조를 파악하고 구현한다. 음원의 주요 검색, 삭제, 추가, 추천 기능을 설계하고 구현한다. 또한 음원을 Item Type, Play Type, Singer Type, Song Type 등이 다양한 클래스로 구분하여, 사용자가 편리하게 음원 관리를 할수 있도록 한다.

(3) 대상

음원관리 프로그램의 관리자는, 충분히 시스템에 대한 정보를 습득하고, 시스템 전반에 대한 이해를 갖춘 사람으로 한정한다. 시스템 문제 발생시 이에 대처할수 있는 충분한 능력을 갖추고 있다고 가정한다.

클라이언트 사용자는 일반 사용자로 한글과 영어를 원활하게 읽고 그 의미를 파악할수 있는 능력을 갖추고 있으며, 화면을 보고 자료를 입력하거나, 버튼을 눌러 창을 열 수 있다고 가정한다. 일반적으로 13세 이상 65세 이하의 사람을 클라이언트 사용자로 가정한다.

(4) 제약 조건

본 명세서에서 언급한 내용을 바탕으로 프로그램을 설계 및 구현한다. 사용되는 자료구조들은 각 항목의 특성에 맞게 개발자가 선택적으로 조정한다. 이외의 사항들은 개발자가 선호하는 방향으로 선택하여 설계 및 구현하되, 단 다음의 항목을 준수한다.

- 시스템 전반의 성능을 향상시키는 방향으로 결정한다.
- 사용자에게 보다 친숙한 방향으로 결정한다.
- 소스코드의 최적화를 통해 메모리의 낭비를 예방한다.
- 소스코드 작성 시 추후 유지보수를 위하여 충분한 주석을 추가한다.

(5) 세부 요구사항

음원관리 프로그램의 출력 화면을 콘솔 창에 띄우는 것에 그치는 것이 아니라, QT Creator 라는 프로그램을 이용하여 사용자를 위한 UI 화면을 디자인하여 구현한다. 또한 사용자가 기입하는 입력 내용이 프로그램에서 요구하는 것과는 다를 경우, 프로그램이 강제로 종료되는 일이 없도록 예외처리에 신경쓰도록 한다.

3. 종합 기술

(1) 정보 흐름

프로그램에 사용자가 넣을수 있는 정보는 크게 두 가지로 가수정보와 음원정보가 있다. 음원정보를 입력하기 위해서는 해당 음원의 가수에 대한 가수정보가 먼저 저장되어 있어야 한다. 가수 정보가 저장되지 않은 가수의 음원은 프로그램에 추가할 수 없다.

사용자는 프로그램에 가수정보(Singer Type - 아이디, 가수명, 팀정보, 성별, 데뷔일자)를 추가하고, 프로그램은 입력받은 가수 정보를 저장한다. 이후 프로그램에 사용자는 음원 정보(ItemType - 아이디, 이름, 작곡가, 연주가, 장르, 태그)를 추가하고, 프로그램은 입력받은 음원 정보를 아이디의 순서에 맞춰서 자동 정렬하여 저장한다.

사용자는 저장된 음원에 한해서, 자유롭게 플레이 리스트를 구성할 수 있다. 저장된 모든 음원 정보들이 나오는 화면에서 플레이 버튼을 누를 경우, 자동으로 플레이 리스트에 음원이 추가되며, 음원이 재생되는 위젯이 등장한다. 플레이 리스트와 뮤직 리스트는 삭제버튼을 통해서 자유롭게 삭제가 가능하며, 클리어 버튼을 통해 전체 삭제 또한 가능하다.

해당 음원 프로그램에는 검색 기능이 존재한다. 이 기능은 음원의 제목을 검색하는 것으로, 음원의 제목을 입력할 경우, 아이디, 가수, 작곡가, 장르, 태그의 정보를 화면에 보여준다. 음원의 전체 제목이 아닌, 일부분만 입력하게 되더라도 해당 음원의 모든 정보가 출력될 수 있도록 하였다.

해당 음원 프로그램에는 추천 기능이 존재한다. 이 기능은 음원을 저장할 때, 음원의 분위기를 Excited, Calm, Dance, 의 네가지로 설정할 수 있도록 하였다. 사용자가 추천 버튼을 누를 경우, 네가지 태그 중 하나를 선택할 수 있다. 음원 리스트에 있는 음원들 중, 선택된 태그에 해당하는 노래들을 화면에 List Table 의 형태로 출력해준다.

(2) 프로세스 기술

번호	1
이름	로그인
입력/출력	입력: 아이디, 비밀번호 (yejinn, yejinn) 출력: 성공시) showmusicwidget.ui 창을 띄운다. 실패시) 에러 메시지가 표시된다.
내용	로그인에 성공한 경우 해당 음원 프로그램을 작동시키는 것이 가능하다.

번호	2
이름	음악 명언 출력
입력/출력	입력: 시작 화면에 있는 ▷버튼을 누른다. 출력: 랜덤으로 음악 명언이 출력된다.
내용	사용자가 프로그램을 처음으로 실행시키는 경우, 첫화면에서 로그인하는 칸과 동시에, 하단에 음악에 관련한 여러 명언들이 랜덤으로 출력되도록 하였다.

번호	3
이름	Add music 버튼 - 뮤직 리스트 - 음원 추가
입력/출력	입력: 화면에서 음원의 정보를 입력. 출력: admin_addmusicwidget.ui 창이 화면에 뜬다.
내용	뮤직 리스트에 음원 정보를 추가한다. 뮤직 리스트에는 가수 리스트에 이미 들어있는 가수에 해당하는 노래만 추가가 가능하다.

번호	4
이름	Add singer 버튼 - 가수 리스트 - 가수 추가
입력/출력	입력: 화면에서 가수의 정보를 입력. 출력: admin_addsingerwidget.ui 창이 화면에 뜬다.
내용	가수 리스트에 가수 정보를 추가한다. (성별, 데뷔일, 팀정보)

번호	5
이름	음원 리스트에서 음원 삭제
입력/출력	입력: DELETE 버튼을 누름. 출력: 화면에 더 이상 해당 음원의 정보가 보이지 않음.
내용	저장된 모든 음원의 정보를 List Table 의 형태로 나타낸 화면에서, 2 열은 DELETE 버튼이 존재한다. 이를 누를 경우, 음원리스트에 있는 해당 음원의 정보가 삭제된다.

번호	6
이름	음원 리스트에서 음원 실행
입력/출력	입력: PLAY 버튼을 누름 출력: playmusicwidget.ui 창이 화면에 뜬다.
내용	해당하는 음원의 간단한 정보를 담은채 NOW PLAYING 이라고 적힌 위젯이 뜨면서, 사용자가 재생중인 음원의 정보를 쉽게 파악할 수 있도록 한다.

번호	7
이름	음원 리스트에서 플레이 리스트에 음원 추가
입력/출력	입력: PLAY 버튼을 누름 출력: PLAYLIST 버튼을 누르고 플레이리스트를 확인하면 해당 음원이 추가되어 있다.
내용	음원 리스트에 존재하는 음원의 정보는 Item Type 이였지만, 이를 Play Type 의 형태로 바꾼 뒤, 플레이 리스트에 해당하는 정보를 넣는다.

번호	8
이름	플레이리스트 출력
입력/출력	입력: PLAYLIST 버튼을 누름 출력: playlistwidget.ui 창이 화면에 뜬다.
내용	플레이 리스트에 들어가 있는 음원들의 정보를 List Table 의 형태로 화면에 띄워준다.

번호	9
이름	플레이 리스트에서 음원 삭제
입력/출력	입력: DELETE 버튼을 누름 출력: 화면에 더 이상 해당 음원의 정보가 보이지 않음.
내용	재생된 모든 음원의 정보를 List Table 의 형태로 나타낸 화면에서, 1 열에는 DELETE 버튼이 존재한다. 이를 누를 경우, 플레이 리스트에 있는 해당 음원의 정보가 삭제된다.

번호	10
이름	음원 리스트의 모든 음원정보 삭제
입력/출력	입력: CLEAR 버튼을 누름 출력: 화면에 더 이상 음원의 정보가 보이지 않음.
내용	“Delete All?” 이라는 메시지 창이 뜨고, OK 를 누를 경우, 음원 리스트에 있는 음원의 모든 정보가 삭제된다.

번호	11
이름	플레이 리스트의 모든 음원정보 삭제
입력/출력	입력: CLEAR 버튼을 누름 출력: 화면에 더 이상 음원의 정보가 보이지 않음.
내용	“Delete All?” 이라는 메시지 창이 뜨고, OK 를 누를 경우, 플레이 리스트에 있는 음원의 모든 정보가 삭제된다.

번호	12
이름	제목 검색 기능
입력/출력	입력: SEARCH 버튼을 누름. 출력: searchwidget.ui 창이 화면에 뜬다.
내용	사용자가 정보를 알고 싶은 음원의 제목을 입력함. 제목을 입력할 경우, 전체 제목이 기억나지 않을 사용자의 상황을 고려하여 제목을 일부분만 검색해도 음원의 정보가 출력되도록 하였다.

번호	13
이름	노래 추천 기능
입력/출력	입력: Recommend 버튼을 누름. 출력: recommend_by_tag.ui 창이 화면에 뜬다.
내용	음원의 정보를 입력할때마다 TAG 로, 현재 음원의 전체적인 분위기를 담는 정보를 받았다. 이를 이용하여 사용자가 원하는 분위기의 태그를 선택하면, 해당 태그를 가진 음원들의 정보를 List Table의 형태로 화면에 보여준다. 사용자를 이를 통해서 비슷한 태그를 가진 음원들의 정보를 파악하고 플레이 리스트를 만드는데에 참고할 수 있다.

(3) 클래스 주요 함수

1. Mainwindow.h (프로그램 시작시 처음에 나오는 창)
    void setmusictexts();
    void on_pushButton_2_clicked(); -> admin_loginwidget.ui 실행
    void on_pushButton_clicked(); -> 음악명언 출력

2. admin_loginwidget.h (로그인 창)
    void on_cancelbtn_clicked();
    void on_loginbtn_clicked();
    void on_idedit_textChanged(const QString &arg1);
    void on_pwedit_textChanged(const QString &arg1);

3. Showmusicwidget.h (사실상 프로그램의 메인 위젯)
    void displayAll(const SortedList<ItemType>&);
    int musicAdd(ItemType inmusic);
    int singerAdd(SingerType insinger);
    int playAdd(Playtype inplay);
    bool getSinger(SingerType& one);
    bool getSong(ItemType& one);
    void on_ListTable_cellClicked(int row, int column);
    void on_search_clicked();

4. admin_addmusicwidget.h (음원 추가 위젯)
    void on_idEdit_textChanged(const QString &arg1);
    void on_nameEdit_textChanged(const QString &arg1);
    void on_composerEdit_textChanged(const QString &arg1);
    void on_singerEdit_textChanged(const QString &arg1);
    void on_comboBox_currentTextChanged(const QString &arg1);
    void on_okbtn_clicked();
    void on_cancelbtn_clicked();
    void on_comboBox_2_currentTextChanged(const QString &arg1);
    ItemType getnewItem() const;

5. admin_addsingerwidget.h
    void on_idEdit_textChanged(const QString &arg1);
    void on_nameEdit_textChanged(const QString &arg1);

void on_teamCombo_currentTextChanged(const QString &arg1);
void on_genderCombo_currentTextChanged(const QString &arg1);
void on_dateEdit_userDateChanged(const QDate &date);
void on_cancelbtn_clicked();
void on_okbtn_2_clicked();

6. playlistwidget.h
void displayAll();
ArrayList<Playtype> getList() const;
ArrayList<Playtype> PlayList;
void on_ListTable_cellClicked(int row, int column);

7. playmusicwidget.h

8. searchwidget.h
void on_titleEdit_textChanged(const QString &arg1);

9. recommend_by_tag.h
SortedList<ItemType> MusicList;
SortedList<ItemType> tempMusicList;
void choosetag(SortedList<ItemType> inlist);
void displayAll();
void on_comboBox_currentTextChanged(const QString &arg1);

(4) ADT 제시

1. Item Type class

```
class ItemType
{
public:
GetID()
```

GetName()
GetComposer()
GetSinger()
GetGenre()
GetsongTag()
SetID(int inID)
SetName(string inName)
SetComposer(string inComposer)
SetSinger(string inSinger)
SetGenre(int inGenre)
SetTag(string inTag)
ReadDataFromFile(ifstream& fin)
WriteDataToFile(ofstream& fout)
RelationType Compare(const ItemType& data)
bool operator==(ItemType data)
bool operator<(ItemType data)
bool operator>(ItemType data)

private:
m_ID
m_Name
m_Composer
m_Singer
m_Genre
m_Tag
}

2. Play Type Class

```
class Playtype
{
    Playtype();
    Playtype(int ID, string SINGER);
    int getId();
    string getSinger();
    string getName();
    void setId(int inid);
    void setSinger(string insinger);
    void setName(string inname);
    bool operator==(Playtype item);
    bool operator>(Playtype item);
    bool operator<(Playtype item);
private:
    int p_ID;
```

```
    string p_Singer;
    string p_Name;
}
```

3. Singer Type Class

```
class SingerType
{
private:
    int s_ID
    string s_Name
    string s_Gender
    string s_group
    string s_Debut
    ArrayList<Song> songList;
public:
    SingerType() {}
    ~SingerType() {}
    int GetID()
    string GetName()
    string GetsGender()
    string GetsGroup()
    string GetsDebut()
    void SetsId(int inid)
    void SetsName(string inName)
    void SetsGender(string inGen)
    void SetsGroup(string ingroup)
    void SetsDebut(string inDebut)
    void SetRecord(int inid, string inname, string ingender, string ingroup, string inde)
    void AddSong(Song insong)
    int ReadDataFromFile(ifstream& fin)
    int WriteDataToFile(ofstream& fout)
    bool operator==(SingerType tempsinger)
    bool operator>(SingerType tempsinger)
    bool operator<(SingerType tempsinger)
}
```

4. User Class

```
class User
{
public:
    User(){}
```

```
    User(string id, string password)
    string getId()
    string getPw()
    int ReadDataFromFile(ifstream& fin)
    int WriteDataToFile(ofstream& fout)
    bool operator==(User temp)
    void setId(string in_id)
    void setPw(string in_pw)
    void setRecord(string in_id, string in_pw)
private:
    string u_id; //user ID
    string u_password;
}
```

5. Song Struct

```
struct Song{
    Song(){}
    Song(int inid, string inname, string intag)
    bool operator==(Song tempsong)
    int id;
    string name;
    string tag;
}
```

6. Sorted List class

```
class SortedList
{
public:
    SortedList();
    ~SortedList();
    void MakeEmpty();
    int GetLength();
    bool IsFull() const;
    bool IsEmpty() const;
    int Add(T data);
    void ResetList();
    int GetNextItem(T& data);
    int Delete(T& data);
    int Replace(T indata);
```

```
    bool Retrieve_Binary(T& inData);
    int GetCurPointer();
    T at(int i);


private:
    T* m_Array;
    int m_Length;
    int m_CurPointer;
}
```

7. Array List Class

```
class ArrayList
{
public:
MakeEmpty()
GetLength()
ResetList()
GetNextItem(T& data)
IsFull()
Add(T data)
IsEmpty()
Delete(T data)
Display()

private:
T m_Array[MAXSIZEE]
m_Length
m_CurPointer
}
```

7. Circular Double Linked Class

```
struct DoubleNodeType
{
    T info
    DoubleNodeType<T>* next
    DoubleNodeType<T>* prev
}
class CircularDoubleLinkedList
{
public:
    CircularDoubleLinkedList()
    ~CircularDoubleLinkedList()
```

```
    void MakeEmpty()
    int GetLength()
    bool IsEmpty() const
    bool IsFull() const
    int Add(T newItem)
    void ResetList()
    void GetNextItem(T& item)
    bool Delete(T data)
    bool Retrieve(T& item)
    T at(int index)
private:
    DoubleNodeType<T>* topPtr
    DoubleNodeType<T>* endPtr
    DoubleNodeType<T>* m_pCurPointer
    int m_length
}
```

4. 초보자 사용 매뉴얼

이 프로그램에 기본으로 설정된 관리자의 이름은 'yejinn' 이고 비밀번호도 'yejinn' 이다. 음원 리스트에는 가수 리스트에 있는 가수에 해당하는 곡들만 추가가 가능하다. 가수 리스트에 없는 가수의 노래라면, 우선 가수 리스트에 해당하는 가수에 대한 성별, 팀정보, 데뷔일의 정보를 추가해야 한다. 이후 음원 리스트에서 노래 추가가 가능하다. 이와 유사하게, 플레이리스트에는 음원 리스트에 있는 곡들만 삽입이 가능하다. 음원 리스트에 존재하지 않는 노래라면, 우선 음원을 추가하면 된다.

프로그램이 시작할 때, 뜨는 화면의 하단에는 음악에 관한 명언들이 랜덤으로 화면에 나온다. 왼쪽에 뜨는 플레이 버튼을 누를 경우, 다른 명언이 출력되는 것을 확인할 수 있다.

음원 리스트에서 플레이 버튼을 누르게 되면, 바로 음원 재생이 가능하고 자동으로 플레이 리스트에 추가된다. 동일한 음원을 두 번 눌러서 재생하게 되면, 플레이 리스트에는 연속으로 두 번 음원이 들어간다.

음원의 삭제를 원할 경우, 음원 리스트에서 DELETE 버튼을 누르면 된다. 모든 음원의 정보를 삭제하고 싶을 경우, CLEAR 버튼을 누르면 된다. 플레이리스트에서도 동일하게 이루어진다.

음원의 검색을 원할 경우 SEARCH 버튼을 누르면 된다. 전체 제목을 입력하지 않고 일부분만 입력해도 검색이 가능하다.

음원의 추천을 원할 경우, RECOMMEND 버튼을 누르면 된다. 원하는 태그를 선택하면 해당하는 음원의 정보가 화면에 뜬다.