**[Project Summary Report](file:///C:/Program%20Files/baidu-translate-client/resources/app.asar/app.html" \l "/#)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Student Id** | 2021329621173 | **Name** | 陈昊暄 | **Phone Number** | | 1935751272 | |
| **Class** | 21计科全英 | **Lab** | Building 3, Computer Laboratory 6 | | **E-mail** | |  |
| **Abstract** | Software Name： |  | | | | | |
| (Please use Android native development technology to design and develop a mobile application software. The data in the software needs to be saved in a local database or server database, or can be provided using third-party open-source web APIs.)  The design and development of this software aims to provide users with an efficient and intuitive tool for detecting pitch and managing music. This software addresses the needs of musicians, music enthusiasts, and educators by offering functionalities such as pitch detection, music playback, and library management. The significance of this software lies in its ability to combine these features into a single, user-friendly application, enhancing the overall music experience. | | | | | | |
| **Date：** | | | | | | |

[**Project Summary Report**](file:///C:/Program%20Files/baidu-translate-client/resources/app.asar/app.html#/#)

Software Name：

Student Id： Name： Class：

**1 Background and significance**

The development of this software is aimed at providing an effective and user-friendly tool for pitch detection and music management on Android devices. With the increasing popularity of mobile applications for music and audio analysis, there is a growing demand for applications that can help users identify musical notes and manage their music libraries efficiently. This software addresses this need by offering pitch detection capabilities alongside robust music management features, making it valuable for musicians, music students, and audio enthusiasts. The significance of this software lies in its ability to enhance musical practice, learning, and enjoyment through accurate pitch detection and convenient music library management.

**2 Requirements Analysis and Database Design**

**User Analysis (UML Use Case Diagram)**

The primary users of this software include:

* **Musicians**: Individuals who play instruments or sing and need a tool to help them identify pitches and manage their music.
* **Music Students**: Students learning music who require assistance with pitch recognition and organization of their practice materials.
* **Audio Enthusiasts**: General users who have a keen interest in audio analysis and music organization.

**Software Requirements:**

* **Pitch Detection**: The software must be able to detect the pitch of sounds captured through the device's microphone and display the corresponding musical note and frequency.
* **Music Management**: Users should be able to scan their device for music files, organize them, and view detailed information about each track.
* **Music Playback**: The application should allow users to play, pause, skip, and stop music tracks within the app.

**Data Storage**

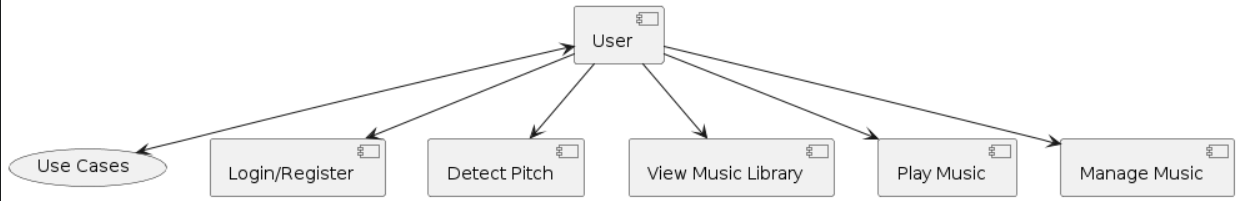
The data in the software needs to be saved in a local database to ensure quick access and offline availability. SQLite is an appropriate choice for the local database due to its lightweight nature and ease of integration with Android applications. The database will store information about the music tracks, such as their titles, artists, durations, and file paths. Additionally, user preferences and settings can be stored in the local database.

**Database Schema:**

* **Table: songlist**
  + songid (INTEGER PRIMARY KEY AUTOINCREMENT): Unique identifier for each song.
  + songname (VARCHAR(100)): The name of the song.
  + songdesc (VARCHAR(50)): The description or artist of the song.
  + songtime (VARCHAR(10)): The duration of the song.
  + songsrc (VARCHAR(200)): The file path of the song.

**3 Software functional module design**

User Analysis (UML Use Case Diagram)



Main Functions of the Software

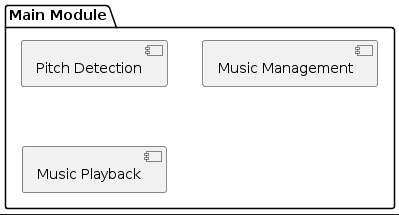
**Pitch Detection**: Detect the pitch of sounds using the device's microphone.

**Music Library Management**: Scan, display, and manage music files on the device.

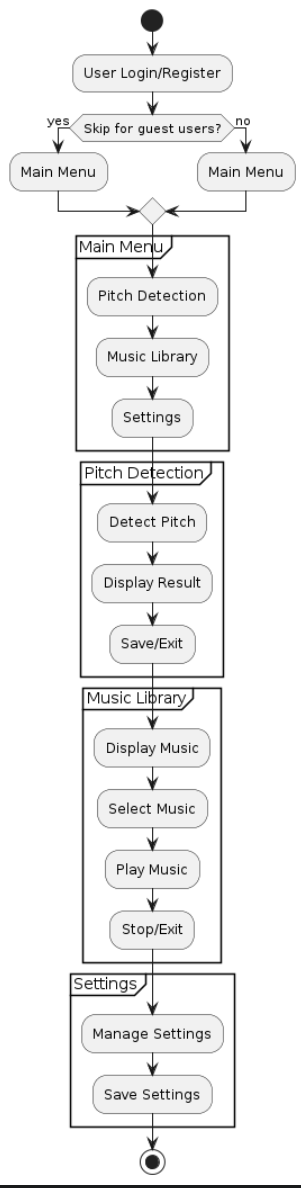
**Music Playback**: Play, pause, and skip music tracks with an integrated player.

**User Interface**: Provide a clean and intuitive interface for ease of use.

Software Functional Module Diagram



Flowchart



Detailed Description of Software Functional Modules

Pitch Detection Module

Function: Detects the pitch of sounds in real-time.

Components: Microphone input, FFT algorithm, pitch display.

Flow:

User starts pitch detection.

The microphone records audio.

FFT algorithm processes the audio signal.

The pitch is displayed on the screen.

User can stop the detection and save the results.

Music Management Module

Function: Manages music files on the device.

Components: File scanner, database manager, music display.

Flow:

User scans for music files.

The app reads metadata from music files.

Music information is stored in a local database.

Music files are displayed in a list.

User can add, remove, or update music information.

Music Playback Module

Function: Plays music files.

Components: Media player, playback controls, progress display.

Flow:

User selects a music file to play.

The media player starts playback.

User can control playback (play, pause, skip).

Playback progress is displayed.

**4 Software Development and Programming**

protected void onCreate(*Bundle savedInstanceState*) {  
 super.onCreate(*savedInstanceState*);  
 setContentView(*R*.*layout*.activity\_music\_list);  
   
 *//初始化数据库  
 SongListDatabaseHelper* dbHelper = new SongListDatabaseHelper(this);  
 *SQLiteDatabase* db = dbHelper.getReadableDatabase();  
  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.hide();  
 }  
  
 recyclerView = findViewById(*R*.*id*.recyclerView);  
  
 *//接收信息查询歌曲  
 String* sql = "SELECT \* FROM songlist;";  
 *List*<*Song*> songList = new ArrayList<>();  
 *Cursor* cursor = db.rawQuery(sql, null);  
 while (cursor.moveToNext()) {  
 *Song* song = new Song();  
 song.setSongName(cursor.getString(cursor.getColumnIndex("songname")));  
 song.setSongDesc(cursor.getString(cursor.getColumnIndex("songdesc")));  
 song.setSongTime(cursor.getString(cursor.getColumnIndex("songtime")));  
 song.setSongSrc(cursor.getString(cursor.getColumnIndex("songsrc")));  
 song.setSongID(cursor.getString(cursor.getColumnIndex("songid")));  
 songList.add(song);  
 }  
 cursor.close();  
  
 *LinearLayoutManager* linearLayoutManager = new LinearLayoutManager(this);  
 recyclerView.setLayoutManager(linearLayoutManager);  
  
 *MusicListAdapter* musicListAdapter = new MusicListAdapter(songList);  
 recyclerView.setAdapter(musicListAdapter);  
}

*@Override*public void onBindViewHolder(*@NonNull MusicListViewHolder holder*, int *position*) {  
 *Song* song = songList.get(*position*);  
 *//绑定信息  
 holder*.nameText.setText(song.getSongName());  
 *holder*.descText.setText(song.getSongDesc());  
 *holder*.timeText.setText(song.getSongTime());  
 *holder*.itemView.setOnClickListener(new *View*.*OnClickListener*() {  
 *@Override* public void onClick(*View v*) {  
 *String* songID=(song.getSongID());  
 *String* songName=(song.getSongName());  
 *String* songDesc=(song.getSongDesc());  
 *String* songTime=(song.getSongTime());  
 *String* songSrc=(song.getSongSrc());  
 *Intent* intent = new Intent(*v*.getContext(), *MusicPlayerActivity*.class);  
 intent.putExtra("songID",songID);*//intent传出数据* intent.putExtra("songName",songName);*//intent传出数据* intent.putExtra("songDesc",songDesc);*//intent传出数据* intent.putExtra("songTime",songTime);*//intent传出数据* intent.putExtra("songSrc",songSrc);*//intent传出数据  
 v*.getContext().startActivity(intent);  
  
 }  
 });

public class *MusicPlayerActivity* extends *AppCompatActivity* {  
 *TextView* songname, songdesc;  
 *ImageButton* prev\_btn, play\_btn, next\_btn;  
 *ImageView* albumArt;  
 *MediaPlayer* mediaPlayer = new MediaPlayer();  
 int songPos;  
 int songID;  
 int songCount;  
 *SQLiteDatabase* db;  
 *Cursor* cursor;  
  
 *@SuppressLint*("Range")  
 *@Override* protected void onCreate(*Bundle savedInstanceState*) {  
 super.onCreate(*savedInstanceState*);  
 setContentView(*R*.*layout*.activity\_music\_player);  
  
 *//载入数据库  
 SongListDatabaseHelper* dbHelper = new SongListDatabaseHelper(this);  
 db = dbHelper.getReadableDatabase();  
   
 *//载入歌曲信息* songname = findViewById(*R*.*id*.songname);  
 songdesc = findViewById(*R*.*id*.songdesc);  
 albumArt = findViewById(*R*.*id*.imageView3);  
 play\_btn = findViewById(*R*.*id*.imageButton2);  
 next\_btn = findViewById(*R*.*id*.imageButton4);  
 prev\_btn = findViewById(*R*.*id*.imageButton3);  
  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.hide();  
 }  
  
 *Intent* intent = getIntent();  
 songID = *Integer*.*parseInt*(intent.getStringExtra("songID"));  
  
 *String* songName = intent.getStringExtra("songName");  
 *String* songDesc = intent.getStringExtra("songDesc");  
 *String* songSrc = intent.getStringExtra("songSrc");  
  
 cursor = db.rawQuery("SELECT \* FROM songlist", null);  
  
 *Cursor* mCount = db.rawQuery("SELECT count(\*) FROM songlist", null);  
 mCount.moveToFirst();  
 songCount = mCount.getInt(0);  
 mCount.close();  
  
 songname.setText(songName);  
 songdesc.setText(songDesc);  
  
 *// 设置专辑封面* setAlbumArt(songSrc);  
  
 try {  
 *Log*.*d*("MusicPlayerActivity", "Trying to play: " + songSrc);  
 if (new File(songSrc).exists()) {  
 mediaPlayer.setDataSource(songSrc);  
 mediaPlayer.prepare();  
 } else {  
 *Toast*.*makeText*(this, "File does not exist: " + songSrc, *Toast*.LENGTH\_SHORT).show();  
 }  
 } catch (*IOException e*) {  
 *e*.printStackTrace();  
 }  
  
 play\_btn.setOnClickListener(new *View*.*OnClickListener*() {  
 *@Override* public void onClick(*View v*) {  
 if (mediaPlayer.isPlaying()) {  
 play\_btn.setImageResource(*R*.*drawable*.ic\_outline\_play\_arrow\_56);  
 mediaPlayer.pause();  
 } else {  
 play\_btn.setImageResource(*R*.*drawable*.ic\_outline\_pause\_56);  
 mediaPlayer.start();  
 }  
 }  
 });  
  
 next\_btn.setOnClickListener(new *View*.*OnClickListener*() {  
 *@Override* public void onClick(*View v*) {  
 playNextSong();  
 }  
 });  
  
 prev\_btn.setOnClickListener(new *View*.*OnClickListener*() {  
 *@Override* public void onClick(*View v*) {  
 playPrevSong();  
 }  
 });  
 }  
  
 private void playNextSong() {  
 songID++;  
 if (songID > songCount) {  
 songID = 1;  
 }  
 songPos = songID - 1;  
 playSongAtPosition(songPos);  
 }  
  
 private void playPrevSong() {  
 songID--;  
 if (songID < 1) {  
 songID = songCount;  
 }  
 songPos = songID - 1;  
 playSongAtPosition(songPos);  
 }  
  
 private void playSongAtPosition(int *position*) {  
 if (cursor.moveToPosition(*position*)) {  
 *//读取歌曲并播放  
 String* songName = cursor.getString(cursor.getColumnIndex("songname"));  
 *String* songSrc = cursor.getString(cursor.getColumnIndex("songsrc"));  
 *String* songDesc = cursor.getString(cursor.getColumnIndex("songdesc"));  
 songname.setText(songName);  
 songdesc.setText(songDesc);  
 mediaPlayer.reset();  
 try {  
 *Log*.*d*("MusicPlayerActivity", "Trying to play: " + songSrc);  
 if (new File(songSrc).exists()) {  
 mediaPlayer.setDataSource(songSrc);  
 mediaPlayer.prepare();  
 mediaPlayer.start();  
 setAlbumArt(songSrc); *// 更新专辑封面* if (mediaPlayer.isPlaying()) {  
 play\_btn.setImageResource(*R*.*drawable*.ic\_outline\_pause\_56);  
 } else {  
 play\_btn.setImageResource(*R*.*drawable*.ic\_outline\_play\_arrow\_56);  
 }  
 } else {  
 *Toast*.*makeText*(this, "File does not exist: " + songSrc, *Toast*.LENGTH\_SHORT).show();  
 }  
 } catch (*IOException e*) {  
 *e*.printStackTrace();  
 }  
 }  
 }  
 *//设定专辑封面* private void setAlbumArt(*String songSrc*) {  
 *MediaMetadataRetriever* mmr = new MediaMetadataRetriever();  
 try {  
 mmr.setDataSource(*songSrc*);  
 byte[] artBytes = mmr.getEmbeddedPicture();  
 if (artBytes != null) {  
 *Bitmap* bitmap = *BitmapFactory*.*decodeByteArray*(artBytes, 0, artBytes.length);  
 albumArt.setImageBitmap(bitmap);  
 } else {  
 albumArt.setImageResource(*R*.*drawable*.background\_gradient); *// 设置默认封面图像* }  
 } catch (*Exception e*) {  
 *e*.printStackTrace();  
 albumArt.setImageResource(*R*.*drawable*.background\_gradient); *// 设置默认封面图像* }  
 }

private void detectPitch() {  
 *//读入buffer李德音频信息* int bufferSize = *AudioRecord*.*getMinBufferSize*(SAMPLE\_RATE, *AudioFormat*.CHANNEL\_IN\_MONO, *AudioFormat*.ENCODING\_PCM\_16BIT);  
 int powerOfTwoBufferSize = getNextPowerOfTwo(bufferSize);  
 short[] buffer = new short[powerOfTwoBufferSize];  
 double[] audioData = new double[powerOfTwoBufferSize];  
 *FastFourierTransformer* fft = new FastFourierTransformer(*DftNormalization*.STANDARD);  
  
 while (isRecording) {  
 audioRecord.read(buffer, 0, bufferSize);  
  
 *// 转换与0填充* for (int i = 0; i < bufferSize; i++) {  
 audioData[i] = buffer[i];  
 }  
 for (int i = bufferSize; i < powerOfTwoBufferSize; i++) {  
 audioData[i] = 0.0;  
 }  
  
 *// 准备FFT，转换数据  
 Complex*[] complexData = fft.transform(audioData, *TransformType*.FORWARD);  
  
 *// 找到峰值频率* double maxMagnitude = -1;  
 int maxIndex = -1;  
 for (int i = 0; i < complexData.length / 2; i++) {  
 double magnitude = complexData[i].abs();  
 if (magnitude > maxMagnitude) {  
 maxMagnitude = magnitude;  
 maxIndex = i;  
 }  
 }  
  
 *// 计算频率  
 // 采样率/缓存大小 = 每个峰值的最小间隔  
 //因为buffer足够小，不考虑极端情况，一段buffer一般只会有一个峰值  
 //峰值索引\*分辨率即为每个峰值的时间间隔即为频率* final double pitch = maxIndex \* SAMPLE\_RATE / powerOfTwoBufferSize;  
 final *String* note = frequencyToNoteName(pitch);  
  
 runOnUiThread(new *Runnable*() {  
 *@Override* public void run() {  
 pitchTextView.setText(*String*.*format*("Pitch: %.2f Hz", pitch));  
 noteTextView.setText(*String*.*format*("Note: %s", note));  
 }  
 });  
 }  
}  
  
*//频率转化为音高*private *String* frequencyToNoteName(double *frequency*) {  
 *String*[] noteNames = {"C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"};  
 *//标准音* double A4 = 440.0;  
 *//音高与频率是对数关系，计算当前频率与标准音的比值，乘12是因为12平均律.  
 //频率没大一倍，音高则高一个八度，一个八度内有12个音符  
 //由于这里是对标准音A4进行对比，A4为第69个音符，故编号时+69  
 //这里最终得到当前时第几个音符* int noteNumber = (int) *Math*.*round*(12 \* *Math*.*log*(*frequency* / A4) / *Math*.*log*(2)) + 69;  
 *//计算是第几个八度* int octave = noteNumber / 12 - 1;  
 *//计算是当前八度的第几个音符* int noteIndex = noteNumber % 12;  
 *//组合* return noteNames[noteIndex] + octave;  
}  
  
*//计算2的N次，以便近似计算log2*private int getNextPowerOfTwo(int *number*) {  
 int power = 1;  
 while (power < *number*) {  
 power \*= 2;  
 }  
 return power;  
}

public void main() {  
 *File* musicDir = *Environment*.*getExternalStoragePublicDirectory*(*Environment*.*DIRECTORY\_MUSIC*);  
 if (musicDir.exists() && musicDir.isDirectory()) {  
 listAllFiles(musicDir);  
 } else {  
 *Log*.*e*(TAG, "Music directory does not exist or is not a directory");  
 }  
}  
  
private void listAllFiles(*File dir*) {  
 *//读取本地歌曲  
 MediaMetadataRetriever* mmr = new MediaMetadataRetriever();  
 *File*[] files = *dir*.listFiles();  
 if (files != null) {  
 for (*File* file : files) {  
 if (file.isFile()) {  
 *String* fileName = file.getName();  
 if (fileName.toLowerCase().endsWith(".mp3") || fileName.toLowerCase().endsWith(".flac") || fileName.toLowerCase().endsWith(".aac")) {  
 *String* filePath = file.getAbsolutePath();  
 try {  
 mmr.setDataSource(filePath);  
 *String* title = mmr.extractMetadata(*MediaMetadataRetriever*.METADATA\_KEY\_TITLE);  
 *String* artist = mmr.extractMetadata(*MediaMetadataRetriever*.METADATA\_KEY\_ARTIST);  
 *String* duration = mmr.extractMetadata(*MediaMetadataRetriever*.METADATA\_KEY\_DURATION);  
 long minutes = *TimeUnit*.MILLISECONDS.toMinutes(*Long*.*parseLong*(duration));  
 long seconds = *TimeUnit*.MILLISECONDS.toSeconds(*Long*.*parseLong*(duration)) - *TimeUnit*.MINUTES.toSeconds(minutes);  
 *String* songTime = *String*.*format*("%d分%d秒", minutes, seconds);  
 db.execSQL("INSERT INTO songlist (songname, songdesc, songtime, songsrc) VALUES (?, ?, ?, ?)",  
 new Object[]{title != null ? title : fileName, artist != null ? artist : "Unknown", songTime, filePath});  
 *Log*.*d*(TAG, "Added song: " + title + " (" + artist + ")");  
 } catch (*Exception e*) {  
 *e*.printStackTrace();  
 }  
 }  
 } else if (file.isDirectory()) {  
 listAllFiles(file);  
 }  
 }  
 } else {  
 *Log*.*e*(TAG, "No files found in directory: " + *dir*.getAbsolutePath());  
 }

**5** [**Summary**](file:///C:/Program%20Files/baidu-translate-client/resources/app.asar/app.html#/#)

1. Background and significance

This software was developed to provide music lovers, teachers and students with a tool to help them detect pitch, manage and play local music files. This software will help users to improve their musical skills and easily organize and enjoy their music collection.

2. Software requirement analysis and design

User requirements: mainly for music lovers, teachers and students, providing pitch detection, music management and playback functions.

Data saving: Use SQLite database to store information of music files, including song name, artist, duration and file path.

3. Function Modules

Pitch Detection Module: Detects pitch in real time using Fast Fourier Transform (FFT) algorithm and displays the corresponding sound name.

Music Management Module: Scan the local music folder, read and save the music file information to the database.

Music Playback Module: plays the selected music file and displays related information, including the album cover.

4. Main Algorithms and Technologies

FFT algorithm: used to extract frequency components from audio signals and determine pitch.

MediaMetadataRetriever: used to extract metadata of music files such as title, artist, duration and album cover.

SQLite database: used to store and manage music file information.

5. Difficulties encountered and solutions

Pitch detection accuracy: when implementing pitch detection, ensuring the accuracy of the detection results is a major challenge. By adjusting the buffer size and using FFT transform, we improved the accuracy of detection.

File Scanning Efficiency: Scanning a large number of local music files can lead to performance problems. By optimizing the file scanning algorithm and using multi-threading techniques, we improved the scanning speed.

Database Management: Ensuring data consistency and integrity is an important task when managing and updating databases. We ensure the security and efficiency of database operations by using transactions and preprocessing statements.

6. Development tools and environment

Development tools: Android Studio

Programming Language: Java

Target platform: Android

7. Future Improvement Direction

Add online music library support: In the future, online music library can be integrated to provide more music resources.

Optimize the user interface: further optimize the user interface to make it more friendly and beautiful.

Improve algorithm performance: Continuously improve the pitch detection algorithm to increase the accuracy and efficiency of real-time detection.

**References**

1. 孙家广, 杨长青.计算机图形学[M].北京：清华大学出版社, 1995.
2. Skolink M I. Radar handbook [M]. New York: McGraw-Hill, 1990.
3. 王鑫, 王斌, 张立明. 基于图像显著性区域的遥感图像机场检测[J]. 计算机辅助设计与图形学学报, 2012, (3):336-337.
4. 王明亮.关于中国学术期刊标准化数据库系统工程的进展[EB/OL].http: //www.cajcd.edu.cn/pub/wml.txt/980810–2.html , 1998–08–16/1998–10–04.
5. Tnesman AM, Gelade G. A feature-integration theory of attention Cognitive Psychology[J]. 1980,12(1): 97-136.
6. OdoniAR.The flow management problem in air traffic control[A].In:OdoniAR,SzegoG,eds.FlowControlofCongestedNetworks[C].Berlin:Springer-Verlag,1987.269-298.