# Wattson: An Innovative IELTS Speaking Assistant Android Application

Kaijie Lai

2034675

Kaijie.Lai20@student.xjtlu.edu.cn

## I. INTRODUCTION

This report presents development and evaluation of "Wattson", an innovative IELTS Speaking Assistant Android application designed to aid students in IELTS speaking practice. The report details the app design, including its user interface and functional features, followed by an evaluation of its effectiveness in aiding IELTS speaking preparation.

### A. Motivation

As a student at Xi'an Jiaotong-Liverpool University, where the majority of students aspire to pursue postgraduate studies abroad, proficiency in the IELTS exam is a common requirement. Among the various sections of the IELTS, the speaking part often poses the greatest challenge for Chinese students.

In my personal experience preparing for the IELTS, I realized that achieving a high score necessitates repetitive practice and training. Recording and reviewing one's own spoken responses is a straightforward and effective method for improvement. However, existing software for oral training on the market are either focused on isolated practice sessions (such as casual conversation practice with a peer) or, like voice assistant apps (e.g. Call Annie), lack targeted training features.

Typically, practice involves the cumbersome process of operating a recorder while simultaneously opening a file of speaking prompts on a computer, selecting a question, and then starting the recording on the phone. Recognizing the inefficiency of this method led me to conceptualize a software solution that combines these two elements. This app aims to streamline the IELTS speaking practice by integrating a database of questions with an easy-to-use recording feature, allowing users to focus on enhancing their speaking skills without the hassle of juggling multiple devices or applications.

### B. Contribution

Generally speaking, I authored most of the code for this project. In this individual report, I will specifically discuss the Record Part and Navigation Feature, highlighting my contributions to these areas. The entire project is designed based on the Model-View-Controller (MVC) architecture in Figure 1. Additionally, I will detail the operational logic of the application, as well as the testing environment, and provide a user guide for understanding and interacting effectively.
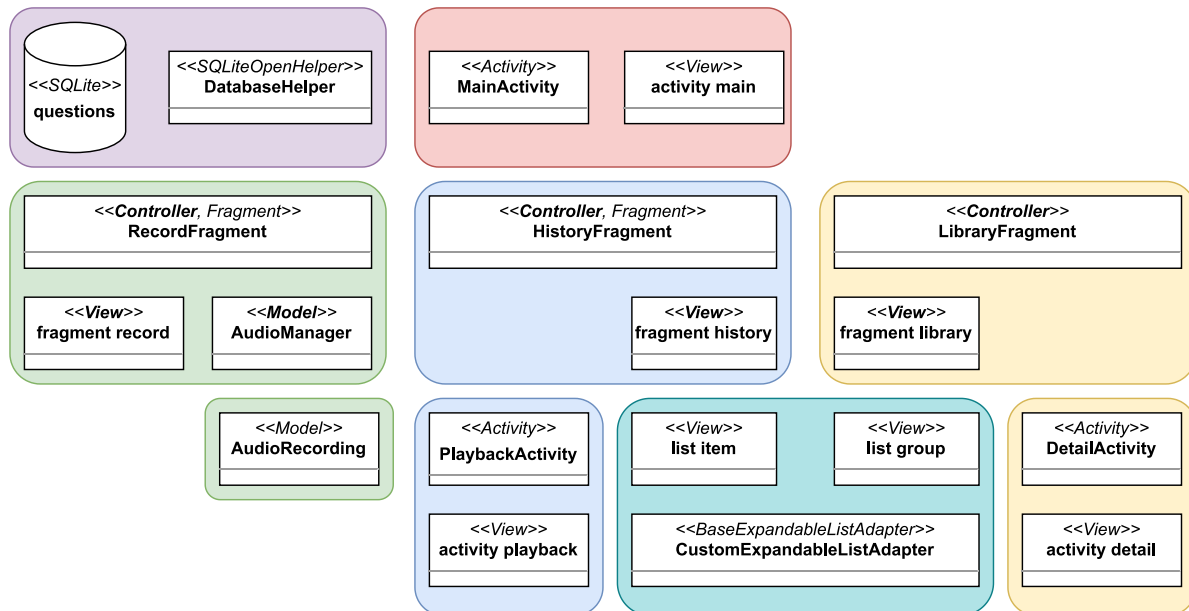


Fig. 1. MVC Architecture of Project

## II. APP DESIGN

In this section, we delve into the intricacies of two primary features of our application: the Record feature and the Bottom Navigation. These features are dissected from multiple perspectives to provide a comprehensive understanding of their design and functionality.

### A. Record Feature

The Record feature is a crucial component of the app, and its design is considered from the following aspects:

1) **User Interface (UI)**: The primary components of the View for the Record feature are defined in the `res/layout/fragment_record.xml`. The corresponding class diagram illustrates the object structure in Figure 4. Initially, the UI layout is as shown in the schematic Figure 2. Interaction and visibility of different UI elements, such as buttons, are controlled in `RecordFragment` by modifying their visibility attributes (visible, invisible, or gone). The View utilizes components like `TextView`, `View`, `SeekBar`, and `ImageButton`, while the ViewGroup includes `FrameLayout`, `LinearLayout`, and `GridLayout`. The app's theme, set in `values/themes/themes.xml`, employs a `NoActionBar` style for a clean interface without a primary bar. Colors used throughout the application are defined in `values/colors.xml`, and all fundamental strings are set in `values/strings.xml`.
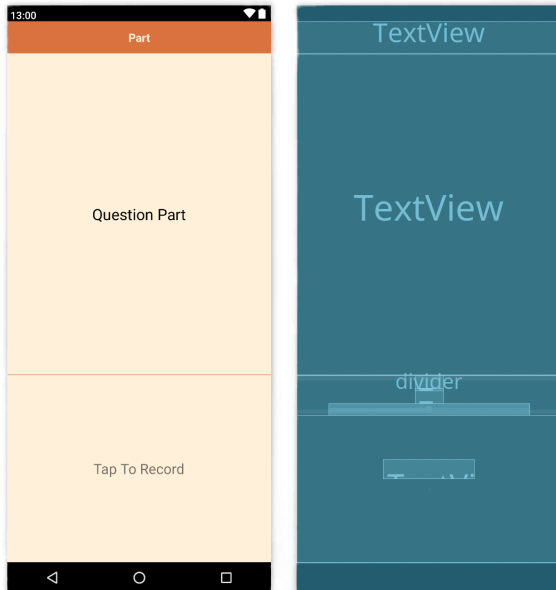


Fig. 2. Layout View of Record Fragment

2) **User Experience (UX)**: Upon opening the app, users are automatically directed to the Record Part (as shown in the attached Figure 4). At the top of the page, users see a question from the database along with its corresponding Part. Following the prompt `Tap to record` begins

the recording process. A subsequent tap changes the prompt to `Tap to pause`, allowing users to pause the recording. Options then include `Tap to resume` or `Long press to finish`. After recording, users can see their recording with options to play it back by pressing the play button and navigate through the recording using the progress bar. The `Next` button allows users to proceed to the next question, saving the current recording. If unsatisfied, the `Delete` option is available to remove the recording, accompanied by a system prompt for confirmation and a toast message `Recording deleted successfully` upon deletion. This workflow constitutes the entire user experience of the Record functionality in the flowchart Figure 3.
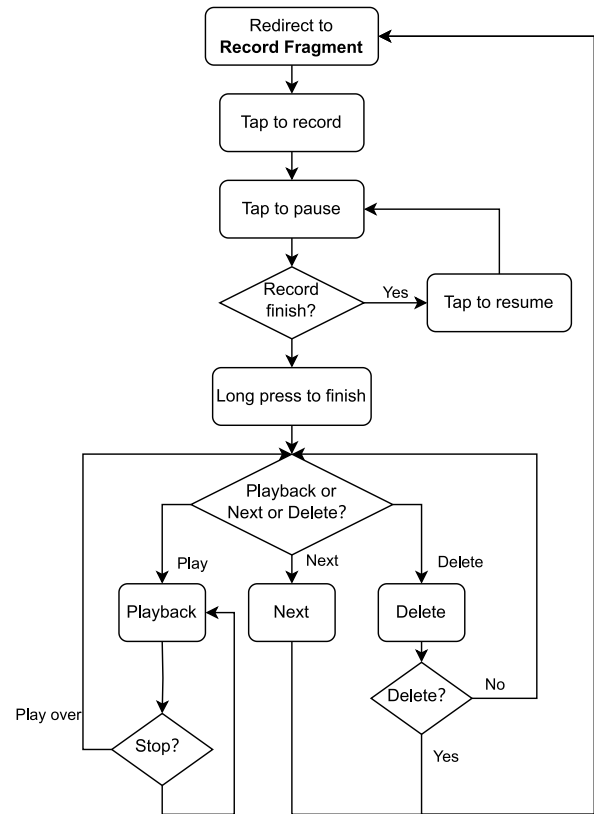


Fig. 3. Flowchart of Record Instruction

3) **Permissions and Data Management**: The `RecordFragment` class manages permissions and data intricately. It checks for audio recording permissions using the `ContextCompat.checkSelfPermission` method and requests them if not already granted. This ensures that the app adheres to Android's security norms. Additionally, the class interacts with the `DatabaseHelper` like in Figure 5 to retrieve and display questions from the database. This involves fetching random questions and their corresponding details, demonstrating the app's ability to manage and

```
        <<View>>
      fragment record
─────────────────────────────────
+ TextView: "@+id/questionPart_part"
+ TextView: "@+id/question_part"
+ View: "@+id/divider"
+ FrameLayout: "@+id/record_part"
   - TextView: "@+id/record_status_text"
   - TextView: "@+id/recording_time_text"
+ LinearLayout: "@+id/seek_bar_layout"
   - TextView: "@+id/playback_time_text"
   - SeekBar: "@+id/seek_bar"
+ GridLayout: "@+id/control_panel"
   - ImageButton: "@+id/play_button"
   - ImageButton: "@+id/next_question_button"
   - ImageButton: "@+id/audio_to_text_button"
   - ImageButton: "@+id/delete_button"
   - ImageButton: "@+id/note_button"
```

```
     <<Controller, Fragment>>
         RecordFragment
────────────────────────────────────
+ audioManager: AudioManager
+ currentRecording: AudioRecording
+ REQUEST_RECORD_AUDIO_PERMISSION: int
+ isRecordingStarted: boolean
+ isPlaying: boolean
+ handler: Handler
+ currentQuestionPart: String
+ currentQuestionTitle: String
+ updateRecordingTimeRunnable: Runnable
+ updateSeekBarRunnable: Runnable
────────────────────────────────────
+ onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState): View
+ setupUI(View view)
+ updateRecordingTime(long millis)
+ updatePlaybackTime(int progress)
+ formatTime(long millis): String
+ onRecordButtonClick(View view)
+ onRecordButtonLongClick(View view): boolean
+ onPlayButtonClick(View view)
+ pausePlaybackIfNeeded()
+ onNoteButtonClick(View view)
+ onTranscribeButtonClick(View view)
+ onDeleteButtonClick(View view)
+ deleteRecording()
+ onNextButtonClick(View view)
+ resetRecordingState()
+ resetPlaybackUI()
+ stopAndReset()
+ onRequestPermissionsResult(int requestCode,
@NonNull String[] permissions, @NonNull int[]
grantResults)
+ displayFromDatabase(View view)
```

```
         <<Model>>
        AudioManager
────────────────────────────────────
+ mediaRecorder: MediaRecorder
+ mediaPlayer: MediaPlayer
+ currentRecordingPath: String
+ context: Context
+ recordingStartTime: long
+ pauseDuration: long
+ lastPauseTime: long
+ isPaused: boolean
+ playbackCompletionListener:
PlaybackCompletionListener
────────────────────────────────────
+ startRecording(String part, String title)
+ stopRecording(AudioRecording
recording)
+ pauseRecording(AudioRecording
recording)
+ resumeRecording(AudioRecording
recording)
+ setPlaybackCompletionListener(Playb
listener)
+ playRecording(String filePath)
+ pausePlayback()
+ resumePlayback()
+ cleanupMediaPlayer()
+ stopPlayback()
+ deleteRecording(String filePath):
boolean
+ getRecordingStartTime(): long
+ getPausedDuration(): long
+ getCurrentPosition(): int
+ getDuration(): int
+ seekTo(int progress)
+ onSeekWhilePaused(int progress)
```

```
        <<Model>>
       AudioRecording
─────────────────────────────────
+ fileName: String
+ filePath: String
+ duration: long
+ isRecording: boolean
+ part: String
─────────────────────────────────
+ getFilePath(): String
+ setDuration(): long
+ isRecording(): boolean
+ getPart(): String
+ setRecording(boolean recordingStatus)
```
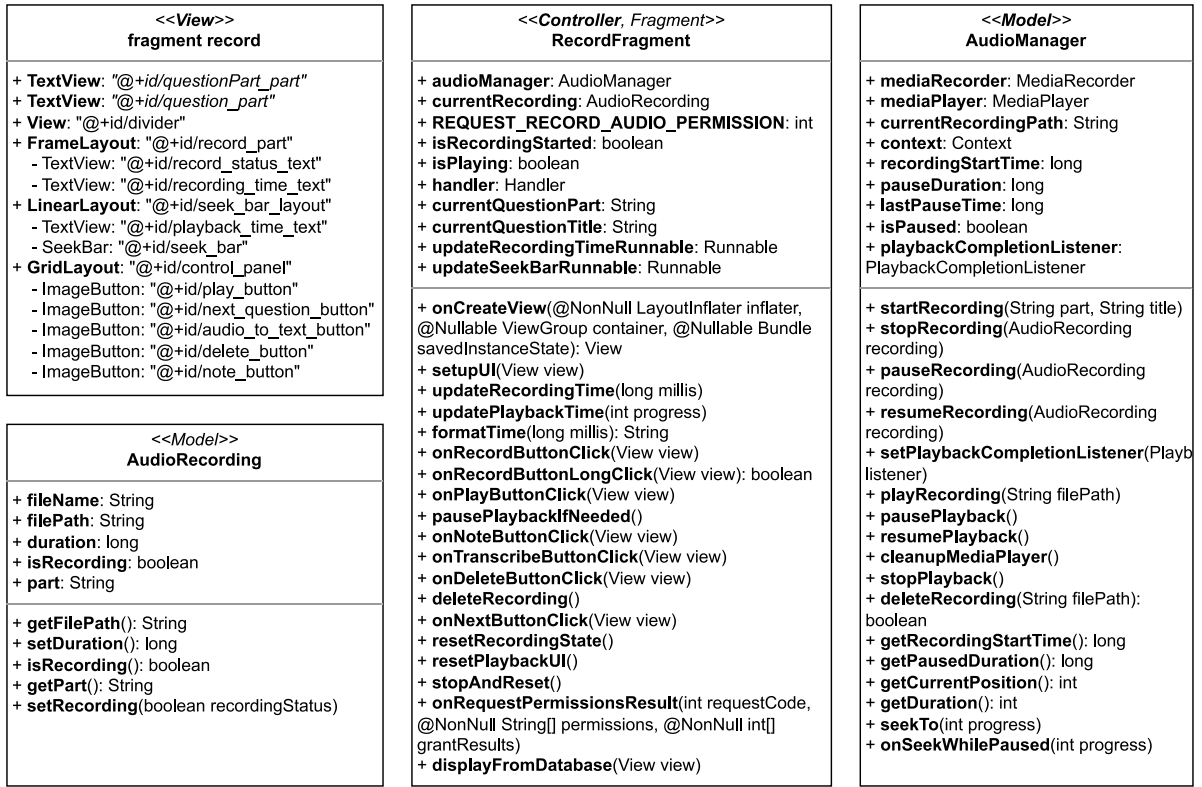
Fig. 4.  Classes Diagram of Record Part

utilize data effectively. The management of audio files, such as creation, playback, and deletion, is handled through the `AudioManager` class, showcasing a well-structured approach to data handling and storage.
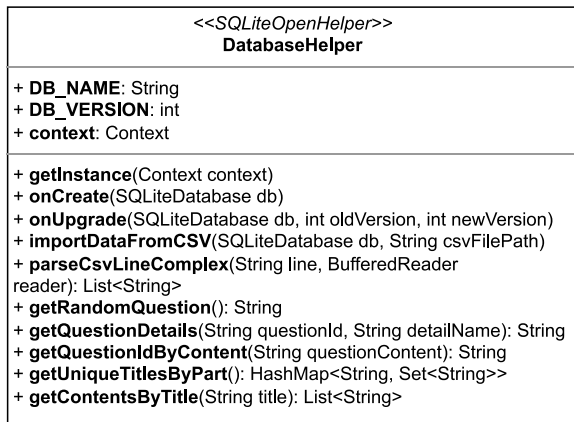
```
        <<SQLiteOpenHelper>>
          DatabaseHelper
──────────────────────────────────────
+ DB_NAME: String
+ DB_VERSION: int
+ context: Context
──────────────────────────────────────
+ getInstance(Context context)
+ onCreate(SQLiteDatabase db)
+ onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
+ importDataFromCSV(SQLiteDatabase db, String csvFilePath)
+ parseCsvLineComplex(String line, BufferedReader
reader): List<String>
+ getRandomQuestion(): String
+ getQuestionDetails(String questionId, String detailName): String
+ getQuestionIdByContent(String questionContent): String
+ getUniqueTitlesByPart(): HashMap<String, Set<String>>
+ getContentsByTitle(String title): List<String>
```

Fig. 5.  Class of Datahelper

4) **Error Handling and User Feedback**: The code demonstrates careful consideration of error handling and user feedback. For instance, if the `Question TextView` or `Part TextView` is null, the app logs an error message, indicating a fail-safe approach to null pointer exceptions. During operations like deleting a recording, the app provides user feedback through `Toast` messages, such as "Recording deleted successfully" or "Failed to delete recording", enhancing the user experience. The use of confirmation dialogs before deleting recordings is another example of thoughtful user interaction design. Moreover, the app logs various actions and errors, such as in `displayFromDatabase` and `onDeleteButtonClick` methods, which aids in troubleshooting and ensures a smooth user experience.

5) **Functionality Implementation**: The implementation of the Record Fragment's functionality is illustrated in the sequence diagram in Figure 6 shown below. This diagram clearly depicts the flow of operations within the Record Fragment, providing a visual representation of the interactions and processes involved. The sequence diagram aids in understanding the sequential steps and the logical flow of the recording feature, from initiating a recording to handling playback and user interactions.

6) **Performance Considerations**: The performance of the application during various stages of interaction in the Record Part is a critical aspect of the user experience. The following Table I presents a detailed analysis of the performance metrics associated with different events. These metrics include the duration of each event in milliseconds, CPU usage percentage, and memory usage in megabytes. The data provides valuable insights into the application's responsiveness and efficiency in resource utilization during key interactions such as starting
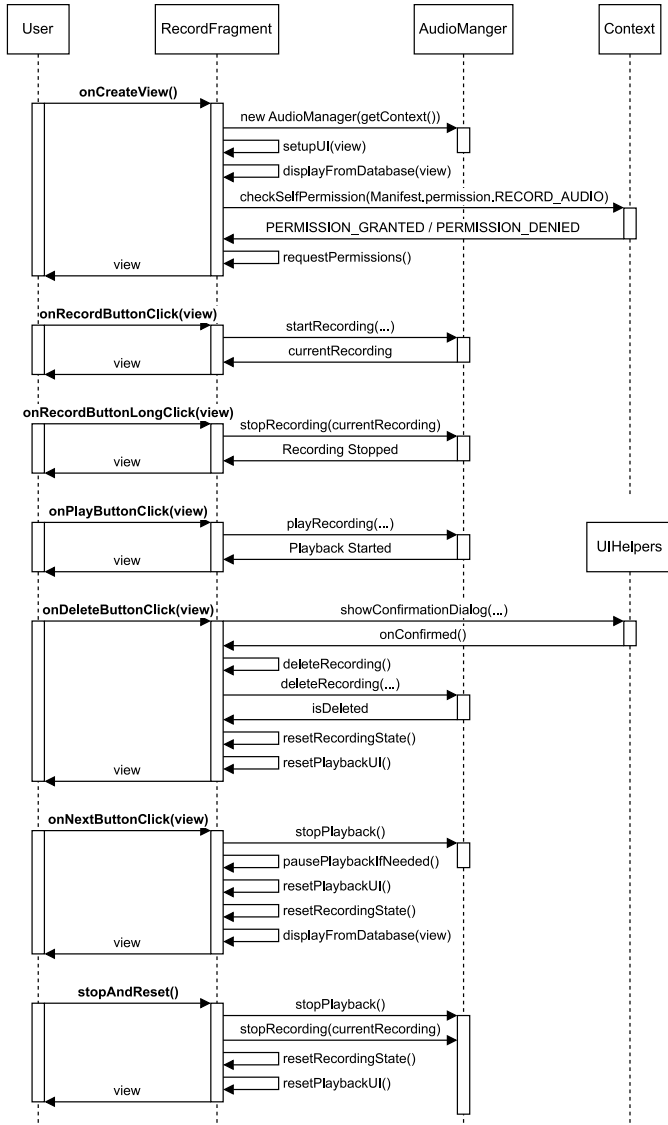
Fig. 6. Sequence Diagram of RecordFragment

and pausing recording, resuming recording, finishing recording, playback, and other functionalities. This analysis is instrumental in identifying potential areas for optimization to enhance the performance of the app.

## III. EVALUATION

### A. Challenges

During the development of our application, several significant challenges were encountered. These include engineering difficulties, user interface concerns, and other issues pertinent to app development. For instance, a major challenge was how to accurately implement the opening and closing of `RecorderMedia` and `PlayMedia`. This issue was critical because improper release of resources could lead to severe crashes, affecting both the stability and reliability of the application.

### B. Solutions

To address these challenges, various solutions were considered. This included the use of `Log` and `Toast` messages during development for situational assessment, as well as the establishment of an `AudioManager` object to specifically manage `RecorderMedia` and `PlayMedia`. Additionally, dedicated `Recording` objects were used for sound management. These measures successfully resolved the issues, enhancing the app's functionality and user experience.

### C. HCI

In terms of Human-Computer Interaction, careful consideration was given to the app's usability. For the `Tap to record` feature, the lower third of the screen was chosen for interaction, as this area is most easily accessible with the thumb, making it convenient for users. This design allows users to start, pause, resume, and stop recording effortlessly, which is highly beneficial for concentrating on their oral practice without the need to constantly look at the phone. After recording, the interface efficiently uses three ImageButtons to execute different functions, making the app more intuitive and user-friendly.

### D. Functionality and Integration

Regarding the question of whether the features of our app work as intended and their synergy with other features, the Record Part has been successfully implemented and integrates well with the overall functionality of the app. However, due to the solo nature of the development process and academic pressures, including language exams and assignments, two features could not be completed in time. These are the manual note-taking feature and the implementation of voice-to-text conversion using Android's native APIs.

| Event | Stage | Duration (ms) | CPU Usage (%) | Memory Usage (MB) |
|---|---|---|---|---|
| Touch Event Press | Start Record | 51.48 | 1 | 49.2 |
| Touch Event Press | Record Pause | 71.09 | 2 | 49.3 |
| Touch Event Press | Resume Record | 91.21 | <1 | 49.2 |
| Touch Event Press | Record Finish | 882.02 | 1 | 49.1 |
| Touch Event Press | Playback | 143.09 | <1 | 49.2 |
| Touch Event Press | Stop Playback | 147.17 | <1 | 49.3 |
| Touch Event Press | Next | 142.03 | 2 | 49.7 |
| Touch Event Press | Delete | 103.32 | 7 | 50.1 |

TABLE I
PERFORMANCE METRICS FOR RECORD PART PROGRESS

*E. Future Enhancements and Reworked Design*

Given more time and resources, several enhancements and reworks are envisioned for the app. Keeping in mind the lecture on context awareness, the following improvements are proposed:

1) **Enhanced Context Awareness:** Incorporating more advanced context-aware features, such as adjusting the difficulty level of questions based on the user's proficiency and learning pace, would make the app more responsive to individual needs.

2) **Interactive Note-Taking Feature:** As previously mentioned, the manual note-taking feature was not completed. With additional time, this feature would be implemented to allow users to jot down notes or key points during their practice sessions, enhancing the learning experience.

3) **Voice-to-Text Functionality:** Implementing the voice-to-text feature using Android's APIs would allow users to see a transcription of their spoken responses, aiding in self-assessment and improvement.

4) **UI/UX Rework:** Based on user feedback and HCI principles, the user interface and experience could be further refined. This might include simplifying navigation, making the app more intuitive, and enhancing visual appeal to keep users engaged.

These proposed enhancements and redesigns are aimed at making the app not only more user-friendly but also more effective as a language learning tool, leveraging the potential of technology to provide a personalized and engaging learning experience.

## IV. CONCLUSION

In conclusion, I successfully navigated through key issues in engineering and interface design, implementing effective solutions that have significantly improved the app's performance and usability. My focus on human-computer interaction principles has resulted in an intuitive and accessible tool for IELTS speaking practice. Wattson stands as a testament to the effectiveness of thoughtful design and user-centered development in creating applications that are not only functional but also user-friendly. I believe that this app will make a substantial difference in assisting users in their language learning journey.

## APPENDIX

*A. Project Repository*

The source code and additional resources for this project are available in the GitHub repository at the following URL: https://github.com/jonlai211/Wattson
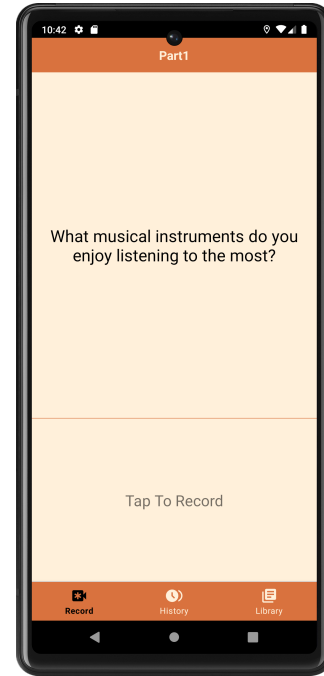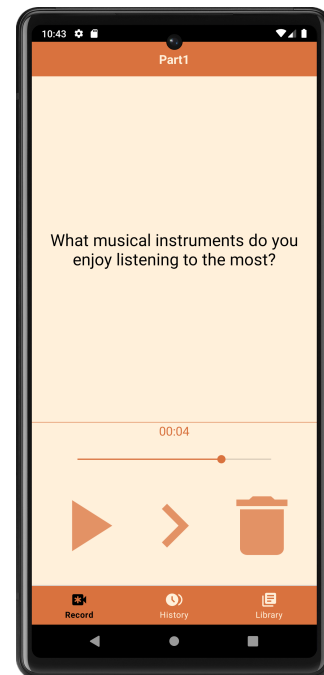
*B. Shortcuts*



Fig. 7.  Record Shortcut



Fig. 8.  Control Shortcut