UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

Final Report

Actor's Line Learning tool

Sudharsan Velraja

sv223@student.le.ac.uk

239042988

Project Supervisor: Dr Karim Mualla
Principal Marker: Dr Anand Sengodan

Word Count:6000
28th April 2025

DECLARATION
All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: sudharsan velraja
Date: 28th April 2025

Contents

1. Introduction

Rehearsal plays a crucial role in the creative journey of an actor, enabling them to enhance their performance, delve into character motivations, and assimilate scripts with greater efficacy [1]. While rehearsals typically involve teamwork with directors, co-actors, and acting mentors, actors often find themselves rehearsing solo, which presents obstacles like insufficient feedback, challenges in sustaining focus, and potential inaccuracies in line delivery [2].

Recent developments in artificial intelligence (AI) have catalysed remarkable progress in the creative arts, encompassing areas such as acting, scriptwriting, and performance coaching [3]. AI technologies now offer resources for self-directed learning and practice, thereby improving individuals' capacity to hone their abilities independently of conventional collaborative settings [4]. Furthermore, advancements in speech recognition and natural language processing (NLP) have proven especially beneficial in automating functions like dialogue simulation and providing feedback on pronunciation [7].

Mobile applications have become significant tools for education and training, providing flexibility, accessibility, and intuitive interfaces [5]. In the realm of acting, mobile rehearsal applications enable users to rehearse scripts anytime and anywhere, thereby enhancing efficiency and minimizing reliance on in-person collaborators. Additionally, the ability to function offline is essential for users with inconsistent internet access, guaranteeing that training resources remain accessible even in areas with limited connectivity [6].

Apropos of these challenges and opportunities, the AI Script Rehearsal App was created. This application allows actors to practice their lines interactively, providing real-time feedback through advanced AI technologies such as speech recognition, semantic similarity evaluation, and text-to-speech functionalities. By integrating a user-focused design with state-of-the-art AI innovations, the app meets the demand for rehearsal solutions that are accessible, self-sufficient, and efficient.

1.2 Challenges Faced by Actors

Actors face multiple obstacles during the rehearsal process, including:

- Securing Scene Partners: It can be quite challenging for students, freelancers, or individuals preparing for auditions on short notice to find someone who is available to rehearse at a suitable time. [8].

- Lack of Constructive Critique: Individual practice sessions often fail to provide valuable insights into performance quality, complicating the process for actors to recognize errors or opportunities for enhancement. [9].

- Motivation and Engagement: Conventional rehearsal techniques often lack variety and do not effectively replicate the vibrant energy of an authentic performance, which can diminish an actor's motivation and hinder their learning outcomes.[10].

- Resource Limitations: Financial constraints may hinder the ability of novice and independent artists to hire professional coaches or invest in specialized rehearsal software. [11].

These challenges underscore the necessity for a rehearsal solution that is more accessible, adaptable, and intelligent.

1.3 Role of AI and Mobile Applications in Creative Arts

Artificial intelligence technologies, including speech recognition, machine learning, and algorithms for assessing sentence similarity, are gaining significant traction in creative fields [12]. Mobile applications serve as robust platforms that broaden access to training resources for a diverse user base [13]. In conjunction with script rehearsal, artificial intelligence can:

- Identify spoken dialogue in real-time, enabling performers to rehearse independently of a human counterpart [14].
- Utilise natural language processing to assess the actor's delivery against the original script, delivering immediate feedback on precision [15].
- Monitor development over time, assisting users in evaluating enhancements and identifying aspects that require focus [16].
- Provide a tailored experience by modifying speech tempo, tone, and vocal characteristics to align with the demands of the character [17].

Mobile devices offer the benefit of portability, allowing users to practice in various locations and at any time, without the requirement for specialised equipment [18].

1.4 Aim and Objectives of the Project

The overarching aim of this project is to design, develop, and evaluate an **AI-powered mobile application** that assists users in **script rehearsal** through **speech recognition**, **real-time feedback**, and **performance tracking**.

The specific objectives are as follows:

- **Develop a user-friendly mobile app** using React Native (Expo framework) to allow platform-independent deployment (Android and iOS).
- **Integrate speech recognition** capabilities to capture the user's spoken lines during rehearsal.
- **Implement a natural language processing (NLP) system** to evaluate the similarity between the spoken lines and the original script, using a lightweight FastAPI server.
- **Design an intuitive, chat-style user interface**, allowing scripts to be displayed like a conversation for ease of rehearsal.
- **Provide visual feedback** after each rehearsal session, including highlighting correct and incorrect lines and visualizing progress over time through charts.
- **Support offline mode**, ensuring that the app remains functional even without continuous internet access.
- **Create adjustable settings** such as speech pitch, rate, and gender to customize the rehearsal experience for different user needs.

Through achieving these objectives, the application aims to empower actors and performers by making rehearsal more accessible, engaging, and effective.

1.5 Structure of the Report

This report is organized into several chapters:

- **Literature Review:** Provides a review of existing literature and technologies related to AI-based rehearsal applications, speech recognition, NLP, and mobile development.
- **System Analysis:** Details the system analysis, including essential functional and desirable requirements and optional functionals.
- **System Design:** Outlines the system design and architecture, including UI/UX considerations.
- **Implementation:** Describes the implementation process, challenges faced, and technical decisions made during development.
- **Testing and Evaluation:** Covers the testing methodologies applied and the evaluation of the system's performance and usability.
- **Results and Discussion:** Presents the results, discusses their implications, and compares the developed application with existing methods.
- **Future Work:** Explores potential future work and improvements.
- **Conclusion:** concludes the report by summarizing the achievements and personal reflections on the project.

2. Literature Review

2.1 Introduction

This chapter examines current technologies, applications, and research pertinent to the creation of a mobile application powered by artificial intelligence for script rehearsal. It addresses available rehearsal tools, progress in speech recognition, natural language processing methods for assessing text similarity, offline-first mobile frameworks, and optimal practices in chat-based user interfaces. The analysis identifies deficiencies in existing solutions and lays the technological groundwork for this project.

2.2 Existing Rehearsal Applications

A variety of mobile applications have been created to aid performers in the process of script learning and rehearsal. Noteworthy examples include:

- Script Rehearser is a mobile application tailored for actors to practice their scripts autonomously. It enables users to record their lines, listen to playback, and rehearse scenes with adjustable pacing and character options [19]. While Script Rehearser provides versatile functionalities such as voice recording and automatic cueing, its main emphasis lies on recording and playback capabilities, in contrast to the AI Script Rehearsal App, which offers real-time feedback on speech accuracy and semantic similarity.

- Rehearsal Pro is a premium application that enables actors to emphasize scripts, insert annotations, and record their performances [20]. It is popular among professional actors due to its user-friendly interface and effective annotation features. Nevertheless, it does not incorporate AI-based feedback systems for assessing the accuracy of spoken lines, semantic validity, or automated rehearsal tracking. Additionally, it does not offer real-time speech recognition or feedback capabilities [21].

- LineLearner is designed to enable users to record their lines and subsequently play them back for rehearsal purposes [22]. While it provides a straightforward and user-friendly interface, it is deficient in advanced functionalities such as speech recognition, real-time feedback, and semantic analysis, which constrains its effectiveness for actors desiring a more comprehensive rehearsal experience.

- Scene Partner: An application dedicated to voice recording and playback, which predominantly utilises pre-recorded audio and lacks AI-driven feedback mechanisms [23].

- ColdRead is a mobile teleprompter application designed to assist actors in delivering their lines during self-taped auditions, although it provides only basic rehearsal tracking functionalities [24].

Although these applications present valuable functionalities like script reading, text highlighting, and audio playback, they lack capabilities such as real-time line recognition, sentence similarity assessment, and an interactive conversational interface. This initiative seeks to fill these deficiencies by integrating real-time speech recognition, AI-driven feedback, and a chat-oriented user interface.

2.3 Advances in Speech Recognition Technologies

Over the last ten years, speech recognition technology has made remarkable progress, primarily driven by advancements in machine learning, deep learning, and the capabilities of mobile hardware. Notable developments encompass:

- The **Google Speech-to-Text** API is a cloud-based service that provides highly accurate transcription capabilities and supports a variety of languages [25].
- **Expo Speech Recognition Module**: A React Native interface that encapsulates native speech recognition APIs, providing a lightweight and offline-capable solution ideal for mobile applications [26].
- **On-device Speech Models**: Contemporary mobile operating systems, including iOS and Android, are progressively enhancing their capabilities for offline speech recognition, thereby augmenting both efficiency and user privacy [27].

The Speech Recognition module from Expo was selected for this project due to its minimal resource requirements, capability for offline operation, and facilitation of seamless cross-platform implementation.

2.4 Natural Language Processing for Sentence Similarity

Evaluating the accuracy of a user's spoken lines requires comparing the spoken input to the original script. In natural language processing, several methods exist to measure text similarity:

- **Cosine Similarity**: Calculates the cosine of the angle between two vectors representing sentences. Suitable for measuring the similarity of bag-of-words representations [28].
- **Levenshtein Distance**: Measures the minimum number of edits (insertions, deletions, substitutions) needed to transform one string into another [29].
- **Semantic Embeddings**: Using models such as Word2Vec, GloVe, or Sentence-BERT to capture semantic meaning beyond exact word matching [30].

Given the need for lightweight and fast evaluation in a mobile context, simpler techniques like Levenshtein Distance or lightweight embedding-based methods are appropriate. This project uses a FastAPI backend to process and score sentence similarity efficiently without overloading the client device.

2.5 Chat-Based User Interfaces for Learning and Engagement

Research indicates that chat-based user interfaces (UIs) can enhance engagement, particularly in educational and training contexts. Benefits include:

- **Familiarity**: Most users are comfortable with messaging apps, lowering the learning curve [31].
- **Turn-Taking Simulation**: Chat UIs mimic real conversations, improving user immersion during dialogue practice [32].
- **Focus on Segmentation**: Presenting script lines as individual chat bubbles helps users focus on one idea at a time, supporting cognitive load theory [33].

By designing the rehearsal app to display scripts in a chat-bubble format, users are guided naturally through scenes, making the experience more dynamic and relatable compared to traditional script readers.

2.6 Offline-First Mobile Application Architecture

An important design requirement for this project is ensuring functionality even when no internet connection is available. Offline-first mobile app design principles suggest:

- **Data Caching and Local Storage**: Storing critical data locally using tools like AsyncStorage or SQLite [34].
- **Graceful Degradation**: Allowing features that depend on network connectivity (e.g., advanced feedback) to degrade gracefully without causing app crashes [35].
- **Sync Mechanisms**: Optionally syncing data when the device is back online [36].

This project leverages AsyncStorage for local script and rehearsal data storage, ensuring seamless rehearsal experiences without requiring constant internet access.

2.7 Identified Gaps and Research Direction

While existing rehearsal applications offer important functionality, they often fall short in the following areas:

- Lack of real-time feedback on spoken lines.
- No integration of sentence similarity evaluation.
- Limited or no offline support.
- Static user interfaces that do not simulate conversation dynamics.

This project aims to fill these gaps by developing an AI-powered, offline-capable mobile application offering real-time rehearsal assistance in an engaging and accessible format.

2.8 Proposed Solution

To address the identified problems and meet the outlined requirements, the proposed solution is the development of a **mobile AI Script Rehearsal Application**

built using **React Native with Expo** for the frontend and **FastAPI** for the backend services (optional for online features). The application will integrate speech recognition and AI-based sentence similarity evaluation to provide real-time rehearsal support.
The core of the solution revolves around a **chat-based rehearsal interface** where users interact with script lines as if messaging a partner. Speech recognition will capture user responses, while a lightweight local database (AsyncStorage) will track rehearsal progress. A simple offline-first design ensures that the application remains functional without relying heavily on internet connectivity.
Key aspects of the proposed solution include:

- **Chat-style Rehearsal Interface**: Scripts are presented in familiar WhatsApp-style bubbles for intuitive interaction.
- **Speech Recognition and Matching**: Spoken lines are processed and evaluated against the original text using semantic similarity scoring.
- **Progress Tracking**: Past attempts are stored and visualized using bar charts and statistics.
- **Offline Capability**: Core rehearsal functions, including script loading, speech capturing, and progress storage, are available offline.
- **Customizable Settings**: Users can adjust speech playback pitch, rate, and gender to tailor their rehearsal experience.

This solution emphasizes **usability**, **accuracy**, and **accessibility**, ensuring that users can rehearse scripts anytime, anywhere, with minimal setup.

3. System Analysis

3.1 Introduction

This chapter offers a comprehensive examination of the system requirements, architecture, and operational workflows pertaining to the AI Script Rehearsal Application. It elaborates on the defined problem statement, as well as both functional and non-functional requirements, accompanied by visual aids such as use case diagrams and system flowcharts. A meticulous system analysis guarantees that the final product meets user expectations and project objectives.

3.2 Problem Definition

Performers, learners, and orators frequently encounter difficulties when practicing scripts in the absence of collaborators. Current applications mainly focus on script reading or line recording, yet they fall short in providing real-time feedback, assessing sentences, and offering engaging, interactive interfaces. Additionally, numerous solutions depend on constant internet access, which restricts their functionality in offline settings.
Consequently, there exists a necessity for a mobile application powered by artificial intelligence that enables users to:

- Engage in independent practice scripts.
- Receive prompt feedback on online precision.
- Function smoothly without internet access.
- Immerse yourself in an interactive, chat-oriented rehearsal setting.

3.3 Requirements

3.3.1 Essential Requirements:

- **Mobile Application**: Develop an intuitive app that acts as the primary interface for users.
- **AI-Powered Text-to-Speech (TTS):** Read scripts aloud with distinct character voices to differentiate roles effectively.
- **Speech Recognition & Error Detection: Analyse** an actor's spoken lines, identify discrepancies, and provide real-time feedback.
- **Semantic Understanding:** Recognize when a spoken line differs from the original text but retains the same meaning, ensuring minor paraphrasing isn't marked as a mistake.
- **Customizable Voice Settings:** Adjust voice gender, pitch, and speed to match user preferences and enhance immersion.
- **Script Management System**: Enable users to upload, store, and organize scripts within the app.
- **Real-Time Corrections:** Offer immediate feedback when an actor makes an error while rehearsing.
- **Deadline & Notifications:** Allow users to set deadlines and receive push notifications to encourage consistent script practice.
- **Video Recording:** Provide an option to record the user's performance for later review.

3.3.2 Recommended Features:

- **Progress Tracker:** Track rehearsal progress over time, offering insights into improvement and consistency.
- **Intuitive User Interface**: Enable users to easily upload, edit, and interact with scripts through a clean, well-organized UI.

3.3.3 Optional Features:

- **Custom Background Colours**: Allow users to personalize script display backgrounds (e.g., yellow or green) to improve readability and reduce eye strain.
- **Full-Script Playback**: Enable users to listen to the entire script with AI-generated voices before practicing.

3.4 Use case Diagram and Workflow

To gain a deeper insight into the interactions between users and the AI Script Rehearsal Application, a use case diagram has been created. This diagram serves to visually illustrate the various methods through which users can engage with the system, such as loading scripts, choosing characters, modifying rehearsal parameters, practicing dialogue with speech recognition, and assessing performance feedback. The use case diagram (Figure 1) offers a comprehensive overview of the fundamental functions of the application.

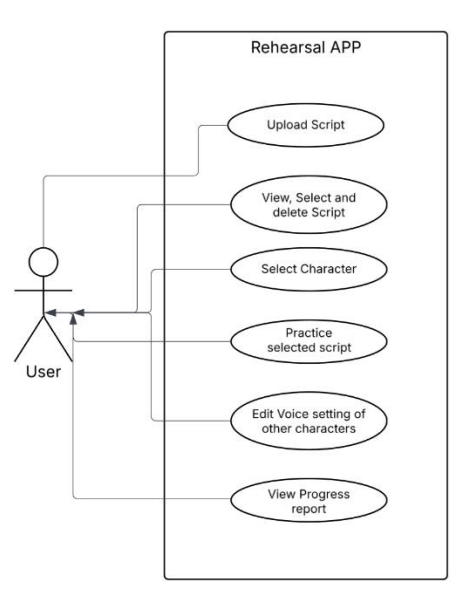

Figure 1: Use case Diagram

3.4.1 System Workflow

The high-level system workflow during rehearsal is as follows:

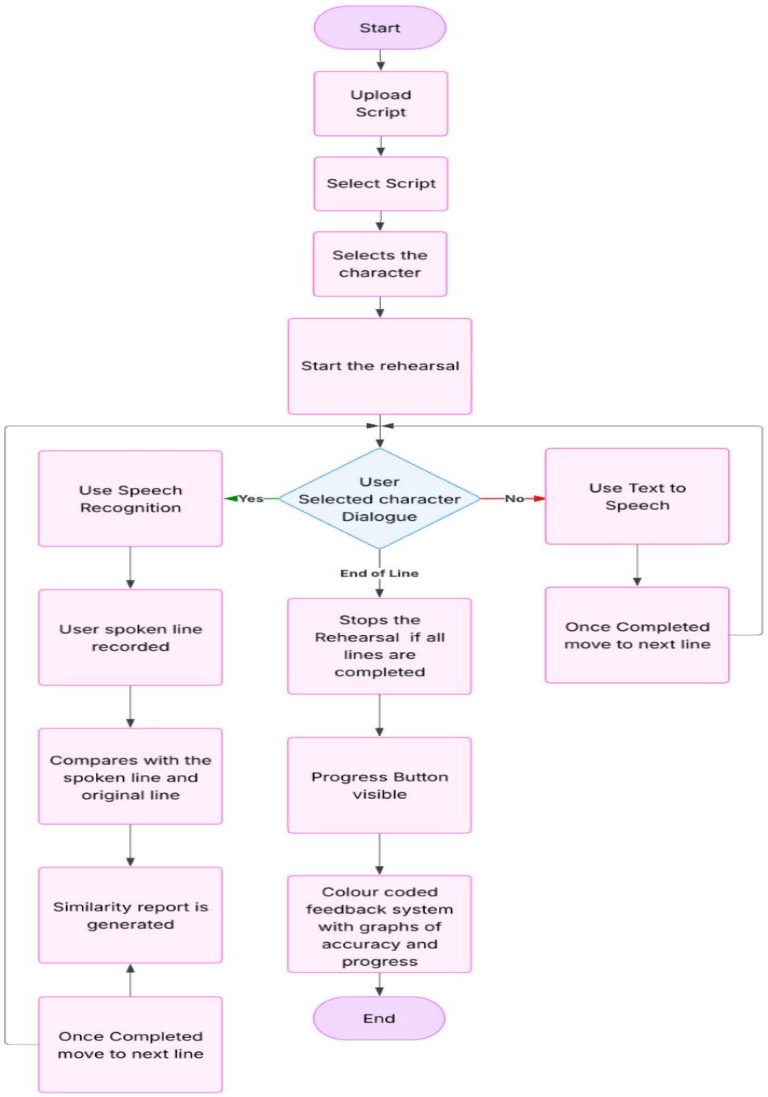

Figure 2: High level System Workflow Diagram

- User upload script documents
- User selects a script and character.
- App displays the script line-by-line in a chat bubble format.
- User taps on a line and speaks.
- User taps the play button to start the rehearsal.
- Speech recognition records the spoken input.
- Spoken input is compared to the original line.
- Sentence similarity score is calculated and stored.
- Colour coded Feedback is shown once the user clicks the progress button.
- After completing all lines, user views a summary with a bar chart showing overall performance.

4. System Design

4.1 Introduction

The process of system design converts the specified requirements and analyses into a comprehensive blueprint for the development of the application. This chapter delineates the architecture, technology stack, module designs, and data flow pertinent to the AI Script Rehearsal Application. Through the use of visual diagrams and thorough explanations of components, it is ensured that the system remains scalable, maintainable, and efficient.

4.2 System Architecture

The application is structured according to a client-server architecture, featuring a frontend developed with React Native Expo and a backend powered by FastAPI. The React Native component manages user interactions, local data storage, speech recognition, and script rehearsal functionalities, whereas the FastAPI backend delivers critical services including text extraction from uploaded PDF documents, character name identification, and sentence similarity assessment utilising sophisticated Natural Language Processing (NLP) models.

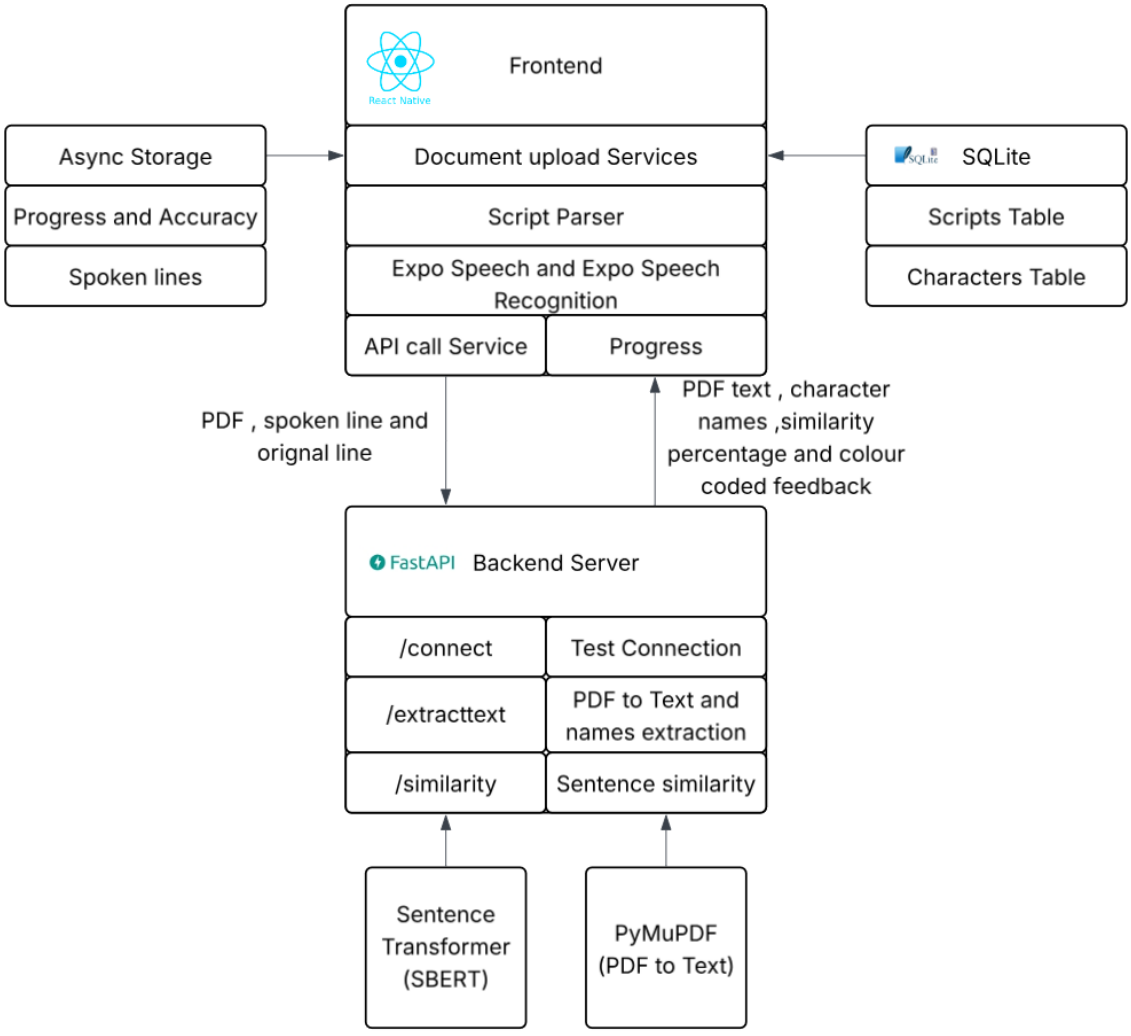

Figure 3: Overall Architecture Diagram of the System

The system follows a **mobile-first architecture** designed for offline-first operation, with optional cloud support for advanced features.

1. Client Side (React Native Expo App)

The client-side architecture involves the **React Native Expo** app, which acts as the interface between the user and the backend server. Key components include:

- **User Interface (UI)**
  - Handles interactions with the user, such as selecting characters, loading scripts, practicing lines, and receiving feedback.
  - Displays scripts, tracks progress, and highlights lines in a WhatsApp-style chat bubble format.

- **Speech Recognition**
  - Converts the user's speech into text using **Expo Speech Recognition**.
  - Tracks real-time progress and compares spoken lines with the original script.
- **Playback and Feedback Display**
  - Provides real-time feedback on the user's spoken lines, such as sentence similarity scoring, and line timing.
  - Displays results like "correct" or "incorrect," and moves through the lines automatically when a line is finished.
- **Local Storage/Progress Tracker**
  - Stores local data like user preferences, script data, and recent progress using **AsyncStorage** and SQLite.
- **Communication with Server**
  - Makes HTTP requests (using **Axios**) to the FastAPI backend for operations such as fetching scripts, extracting character names and processing feedback.
  -

2. Server Side (FastAPI Backend)

The server side is built with python **FastAPI**, which handles requests from the client-side app. The backend architecture includes the following components:

- **FastAPI Application**
  - Handles HTTP requests, routing, and integrates with other backend components. CORS middleware is enabled on the server to allow requests from any origin.
  - Provides endpoints to handle script retrieval, line-by-line feedback, and storing user progress.
- **AI Processing Engine**
  - Integrates AI models to compare spoken lines with the script and provide feedback on accuracy.
  - Uses **Sentence Transformers** (bert-base-nli-mean-tokens) for semantic similarity
- **API Endpoints**
  - Provides APIs for interacting with the client-side app, including:
  - **POST /similarity**: Submit feedback for a given line based on speech recognition.
  - **POST /extracttext**: Extract text from PDF and character names and send them as response.
  -

3. Communication Between Client and Server

- **HTTP Requests**: The client communicates with the server via HTTP requests, typically using **RESTful APIs** (with JSON payloads). Common interactions include:
  - Fetching scripts and rehearsal data.
  - Submitting speech input for real-time processing and feedback.

4. Data Flow

- **Client to Server**:
  - Request sent to the server for sentence similarity scoring
  - Response sent back with feedback.
- **Server to Client**:
  - Server sends script text data to the client
  - Client displays the script and allows for rehearsing

4.3 Technology Stack

In the development of the AI-driven script rehearsal application, a diverse array of technologies was chosen to guarantee scalability, efficiency, and a smooth user experience. The technologies employed encompass the frontend, backend, and database management, with each fulfilling a distinct function within the comprehensive system architecture. The chosen technologies for this initiative comprise React Native Expo, FastAPI, SQLite, Speech Recognition, and Sentence Similarity Scoring.

Layer	Technology	Purpose
Frontend	**React Native (Expo)**	**Mobile App Development, develop services like Script parser to extract dialogue, actions and description from scripts**
Speech API	**Expo Speech & Expo Speech Recognition**	**Text-to-Speech (TTS) and Speech-to-Text (STT) functionalities**
Local Storage	**AsyncStorage and SQLite**	**Store user progress, Script data, character list.**
Backend (Optional)	**FastAPI (Python)**	**Sentence similarity computation using pre-trained SBERT models, Extract text from PDF, Extract character names from scripts**
Charts	**React native charts**	**Display bar charts for visualizing user progress**
State Management	**React Native Hooks/Context**	**Manage app state across components**

Table 1: Technology stack list

1. React Native Expo (Frontend)

- **React Native, an open-source framework created by Facebook, empowers developers to construct cross-platform mobile applications utilizing JavaScript and React. This framework facilitates the creation of applications that function smoothly on both iOS and Android platforms, eliminating the necessity for distinct codebases for each. Expo comprises a collection of tools designed to enhance React Native, offering a user-friendly framework for the development and deployment of React Native applications [37].**

- **Role in the App:**

- **User Interface (UI): React Native Expo allows for the creation of a responsive, mobile-friendly UI that displays scripts, tracks progress and manages user interactions such as character selection and playback of lines.**
- **Speech Recognition Integration: Expo's integration with libraries like Expo Speech Recognition [41] allows the app to capture user speech, convert it into text, and compare it to the script.**

2. FastAPI (Backend)

- **FastAPI is an advanced and high-performance web framework designed for creating APIs using Python 3.7 and later, leveraging standard Python type hints. It is recognized for its rapid execution and user-friendly interface, facilitating automatic data validation, documentation, and serialisation. FastAPI is constructed upon Starlette for web functionalities and Pydantic for data validation.**

- **Role in the App:**
- **Backend API: FastAPI handles all the API requests from the client-side application. It processes requests related to extracting text from PDF files, sentence similarity checking, character extraction from script text.**
-

3. SQLite (Database)

- **SQLite is a relational database engine that operates without a server, is self-sufficient, and requires no configuration [40]. It is particularly well-suited for embedded systems or applications that require the storage and querying of small to moderate volumes of data.**

- **Role in the App:**
- **Data Storage: SQLite stores the user progress, rehearsal data including script file, text, character name list and their voice settings ensuring that data can be easily accessed and updated by the app during rehearsals.**
- **Offline Access: Since SQLite is a lightweight database, it allows the app to work offline.**

4. Speech Recognition

- **Speech recognition technology facilitates the transformation of verbal language into written text [41], [42]. In this application, speech recognition is employed to record the actor's dialogue during practice sessions and juxtapose it with the original script.**

- **Role in the App:**
- **Voice Input: The app listens to the user's speech input, converting it into text using speech recognition libraries like Expo Speech Recognition [41].**
- **Feedback Generation: This technology is essential for providing real-time feedback on pronunciation accuracy, line timing, and sentence similarity.**

5. Sentence Similarity Scoring

- **Sentence Similarity Scoring is a method employed to assess the degree of similarity between two textual segments [43], [44]. In this application, it is utilized to juxtapose the transcribed lines spoken by the actor against the original script, offering insights into the extent of their correspondence.**

- **Role in the App:**
  - **Feedback Accuracy: Sentence similarity scoring helps the system assess how well the user has delivered the lines, accounting for minor differences in phrasing.**

4.4 Database Design

The application uses AsyncStorage and SQLite to store user progress and scripts locally, allowing the rehearsal app to function without continuous server communication.

4.4.1 Local data storage using SQLite

The AI Script Rehearsal App uses a local SQLite database named localdb to manage and store rehearsal data efficiently. The database consists of two primary tables:

- **scripts: Stores information about uploaded script files and user-selected characters.**
- **characters: Stores associated characters for each script, including optional voice settings.**

The structure has been designed to ensure fast retrieval, update operations, and extensibility for future features such as advanced voice customization and rehearsal tracking.

The Figure 4 below shows entity relationship diagram of both the primary tables.

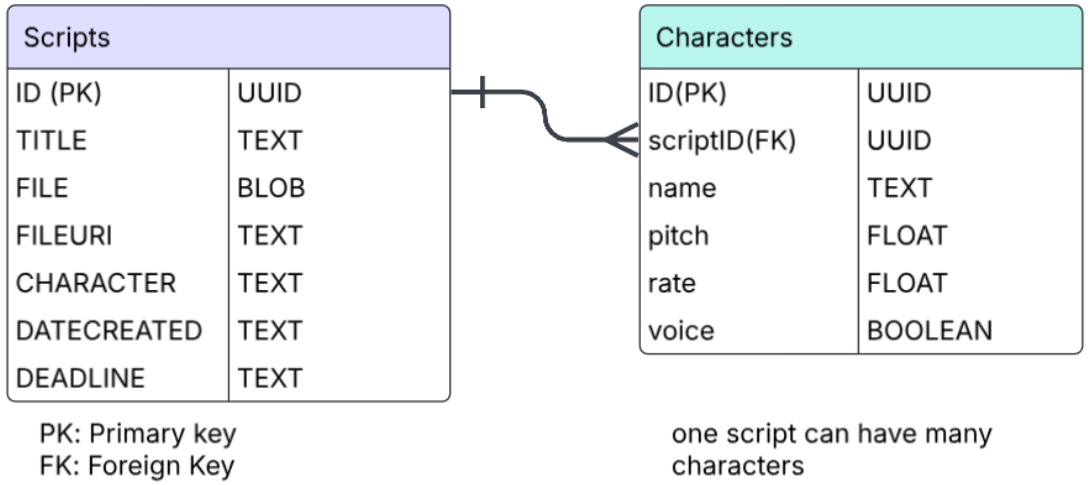

Figure 4: SQLite Database ER Diagram

Column Name	Type	Description
id	**UUID (PK)**	**Unique identifier for each script.**
title	**TEXT**	**Title of the script.**
file	**BLOB**	**Binary storage for the script file.**
fileURI	**TEXT**	**Local URI to the file location.**
content	**TEXT**	**Parsed text content from the file.**
character	**TEXT**	**Name of the user-selected character.**
dateCreated	**TEXT**	**Timestamp when script was created.**

deadline	**TEXT**	**User-defined deadline for rehearsal.**

Table 2: Field Description of scripts Table

Column Name		**Type**	**Description**
id		UUID (PK)	Unique identifier for each character.
scriptID		UUID (FK)	References the script to which it belongs.
name		TEXT	Character name.
pitch		FLOAT	Custom pitch setting for the voice.
rate		FLOAT	Custom rate setting for the voice.
voice		BOOLEAN	Gender true for female , false for female

Table 3: Field Description of Characters Table

4.4.2 Local data storage using AyncStorage

AsyncStorage is utilized to handle simple local data persistence. AsyncStorage is a key-value storage system available in React Native that allows the app to store small amounts of data on the device, even after the app is closed or restarted [45]. This solution was chosen over heavier options like SQLite because the amount of data is small, the structure is simple.

- In this project, AsyncStorage is used primarily to:
  - Store the user's rehearsal progress, including Spoken lines and original lines.
  - Progress percentages over the last five rehearsal attempts.
- Enable quick access to recent rehearsal results without requiring an internet connection.

By leveraging AsyncStorage, the application ensures that users can track their performance history smoothly and efficiently without relying on a backend server for every session. This provides an offline-first experience, which is critical for usability in various environments where network connectivity may be inconsistent.

4.5 Backend Design

The backend architecture for this initiative was constructed utilising FastAPI, a contemporary web framework in Python recognized for its rapid performance, development simplicity, and automatic generation of documentation. This backend is tasked with managing client requests, processing uploaded PDF documents, extracting pertinent information, and delivering structured responses for integration within the mobile application.

4.5.1 Technology Stack

Framework: FastAPI
Programming Language: Python 3.11

- Libraries:
  - PyMuPDF (fitz): For reading and extracting text from PDF documents.
  - Pydantic: For data validation and serialization.
  - re (Regular Expressions): For text processing and pattern matching.

- Sentence Transformers (BERT-based): for comparing two sentences and to find the similarity.
- Middleware:
- CORS Middleware: Configured to allow cross-origin requests, enabling seamless communication between the frontend (React Native app) and the backend server.

4.5.2 API Endpoints

The backend exposes two primary RESTful API endpoints:

1. **POST /connect**

a. Purpose: Establishes a basic connection to verify the server is reachable.
b. Request Body: Accepts a JSON object containing a name field.
c. Response: Returns a simple message confirming the connection with the provided name.

2**.POST /extracttext**

- Purpose: Accepts a PDF file upload, extracts the text content, and identifies potential character names from the script.
- Request: Multipart/form-data containing a PDF file.
- Processing Steps:
- Reads the uploaded PDF file into memory.
- Extracts textual content from the PDF using PyMuPDF.
- Parses the extracted text to detect possible character names based on scriptwriting conventions (e.g., uppercase names before dialogues, names ending with a colon).
- Response: Returns the extracted text along with a list of detected character names.

4.5.3 Text and Character Extraction Logic

The backend implements custom logic for parsing scripts, designed specifically for typical screenplay formatting:

- **Text Extraction:**
  - Utilizes PyMuPDF to iterate through all pages in the PDF and aggregate their text content.
- **Character Name Detection:**
  - Names are identified based on formatting rules:
  - Lines entirely in uppercase.
  - Lines ending with a colon (:) that are in uppercase.
  - Filtering includes:
  - Ignoring lines longer than 40 characters.
  - Excluding common scene headings (e.g., "INT", "EXT", "DAY", "NIGHT").
  - Allowing only one or two-word names.
  - Removing artifacts like "(CONT'D)" that commonly appear in screenplays.

  This lightweight heuristic method offers high accuracy for typical English-language scripts, enabling reliable automatic extraction of dialogue characters for rehearsal purposes.

4.3.4 Sentence Similarity API

The backend includes a dedicated /similarity endpoint designed to compute the semantic similarity between two sentences. This functionality supports real-time performance feedback during script rehearsals.
When a POST request is made with two sentences, the server follows these steps:

- **Sentence Embedding:**
  The input sentences are encoded into high-dimensional vectors using a pre-trained **Sentence-BERT (SBERT)** model [4]. SBERT generates dense sentence representations that capture semantic meaning beyond simple keyword matching.
- **Cosine Similarity Computation:**
  Cosine similarity measures the angle between the two sentence vectors, calculating their semantic closeness. The formula is:
- **Similarity Percent Scaling:**
  The cosine similarity value (ranging from 0 to 1) is scaled into a percentage to improve user interpretation.

- **Colour-Based Feedback:**
  Depending on the similarity percentage, a color code is returned:
  - Green: similarity ≥ 90%
  - Orange: 70% ≤ similarity < 90%
  - Red: similarity < 70%

This colour coding enables immediate visual feedback for users during rehearsals.

- **Error Handling:**
  - Robust exception handling ensures the API returns informative error messages in case of failure during encoding or computation.
  - Thus, the Sentence Similarity API forms a critical part of the backend system for providing automated, real-time rehearsal evaluation.

4.5.4 Middleware and Security

- **CORS Configuration:**
  - Configured to allow all origins ("*"), all methods, and all headers.
  - This ensures that the mobile application can interact with the backend without encountering cross-origin request errors, simplifying development and deployment.
- **File Handling:**
  - The backend directly processes the uploaded files in memory without persisting them to disk, reducing storage overhead and enhancing security.

4.6 Frontend Design

The wireframes were created using Figma to visualize the app's structure, navigation flow, and user interaction with key features such as script upload, script playback, progress tracking, and settings management.

4.6.1 Wireframe Overview

The Figure 5 below shows the different wireframe screens that’s been designed in FIgma.

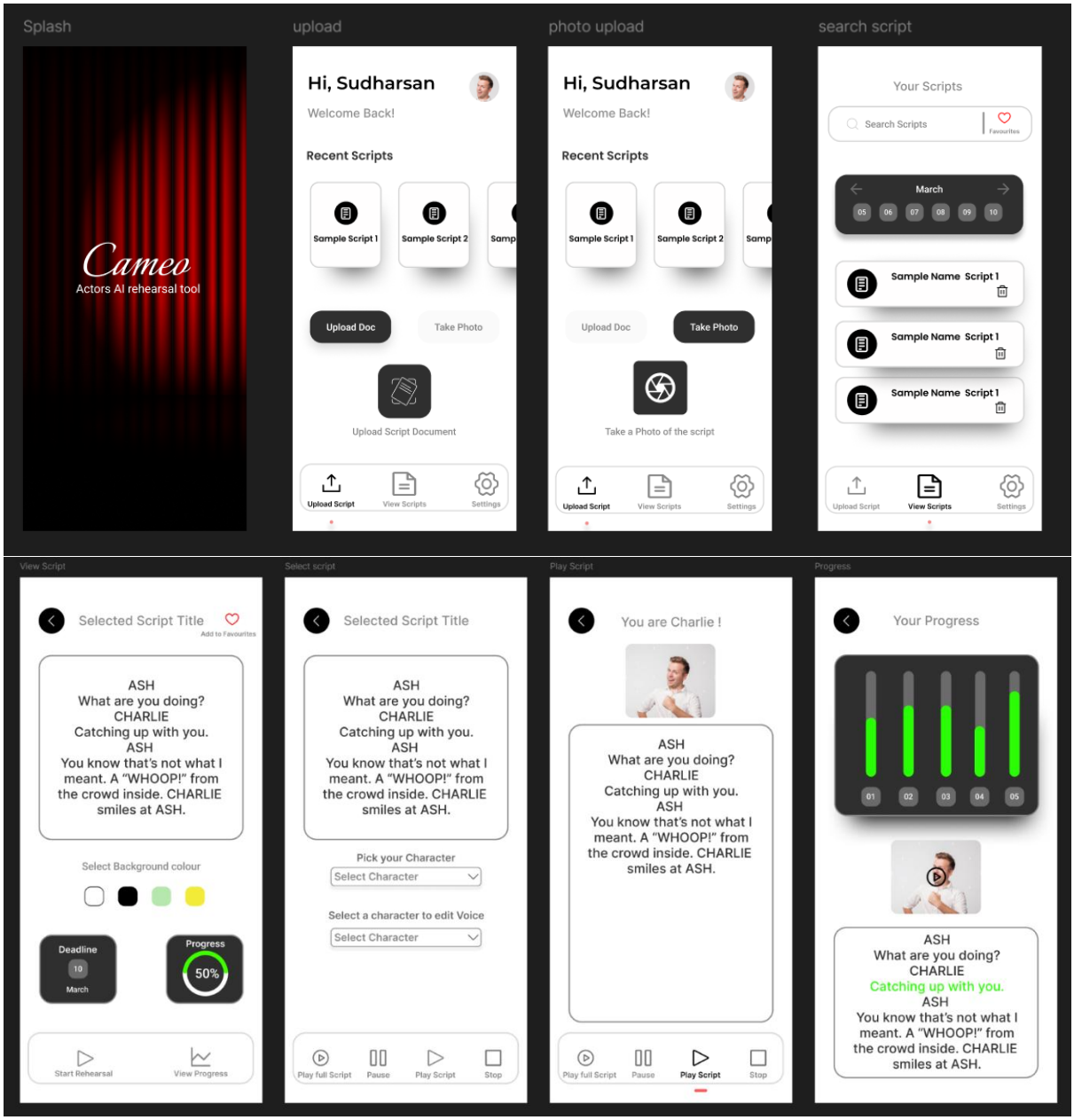

Figure 5: Wireframe Screen design

The wireframe consists of the following screens:

1. Splash Screen
    a. Displays the app name ("Cameo") and tagline ("Actors AI rehearsal tool") with a visual background of theatre curtains, setting the theme of acting and performance.
2. Upload Screen
    a. Greets the user personally ("Hi, Sudharsan").
    b. Shows recent scripts in a horizontally scrollable format.
    c. Options to upload a document or take a photo of a physical script.
    d. Navigation bar with three options: Upload Script, View Scripts, Settings.
3. Photo Upload Screen
    a. Similar layout to Upload Screen but focuses on script upload by taking a photo.
4. Search Scripts Screen

a. Allows users to search for existing scripts by name.
b. View favorite scripts and browse scripts by date (calendar layout).
c. Lists available scripts with easy access for rehearsal.

5. View Script Screen
   a. Displays the selected script with character dialogues.
   b. Option to set background color for better readability.
   c. Button to start script rehearsal and a progress tracker showing completion percentage.
6. Select Script Screen
   a. Enables users to pick a character to rehearse.
   b. Users can also select a character to edit voice parameters (e.g., pitch, rate).
7. Play Script Screen
   a. Displays the script during rehearsal.
   b. Buttons to Play Full Script, Pause, Play Script, and Stop rehearsal session.
8. Progress Screen
   a. Shows a bar chart of user's past 5 rehearsal attempts
   b. Displays user's spoken dialogues compared to original script, highlighting spoken and missed words.

5. Design Implementation
6. Testing and Evaluation
7. Results and Discussion
8. Conclusion

9. References

[1] D. McGaw, "The Importance of Rehearsal for Actors," *Backstage Magazine*, 2020. [Online]. Available: https://www.backstage.com/magazine/article/importance-actors-rehearsal-tips-67578/. [Accessed: 26-Apr-2025].

[2] S. Campbell, 'Can you practice acting on your own?', *StageMilk*, 26-Jan-2012. [Online]. Available: https://www.stagemilk.com/can-you-practice-acting-on-your-own/. [Accessed: 26-Apr-2025].

[3] P. Lam, 'The Impact Of Artificial Intelligence On The Art World', *Forbes.com*, 02-Feb-2024. [Online]. Available: https://www.forbes.com/councils/forbesbusinesscouncil/2024/02/02/the-impact-of-artificial-intelligence-on-the-art-world/. [Accessed: 26-Apr-2025].

[4] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial Neural Networks: A Tutorial," *Computer*, vol. 29, no. 3, pp. 31-44, Mar. 1996.

[5] M. Vukovic, "The Role of Mobile Apps in Education," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 11, no. 4, pp. 65-73, 2017.

[6] P. Gorla and V. Merugu, "Offline Mobile Applications: Benefits and Challenges," *International Journal of Computer Applications*, vol. 120, no. 2, pp. 20–25, 2015.

[7] Google AI, "Speech-to-Text: Automatic Speech Recognition," Google Cloud, 2024. [Online]. Available: https://cloud.google.com/speech-to-text . [Accessed:26-Apr-2025].

[8] L. Ramirez, "Round-Up: 5 Rehearsal Problems… and Solutions!" *Theatrefolk*, [Online]. Available: https://www.theatrefolk.com/blog/round-5-rehearsal-problems-solutions. [Accessed: Apr. 27, 2025].

[9] D. Genik, "How To Come Back To Rehearsal Improved And Better," *David Genik*, [Online]. Available: https://www.davidgenik.com/blog/how-to-come-back-to-rehearsal-improved-and-better.

[10] "The Challenges of Acting | Common Challenges for Actors," *StageMilk*, [Online]. Available: https://www.stagemilk.com/the-challenges-of-acting/. [Accessed: Apr. 27, 2025].

[11] "Experiencing Theatre: Challenges Actors Face in Bringing," *Course Sidekick*, [Online]. Available: https://www.coursesidekick.com/english/1682941. [Accessed: Apr. 27, 2025].

[12] M. McCormack, "Artificial Intelligence in Creative Industries," *AI & Society*, vol. 37, no. 2, pp. 645–658, 2022.

[13] S. Wang, C. Yu, and Z. Chen, "Democratizing Artificial Intelligence Through Mobile Platforms," *IEEE Access*, vol. 9, pp. 112345–112355, 2021.

[14] A. Graves et al., "Speech Recognition with Deep Recurrent Neural Networks," *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.* (ICASSP), 2013, pp. 6645–6649.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proc. NAACL-HLT*, pp. 4171–4186, 2019.

[16] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *arXiv preprint* arXiv:1706.05098, 2017.

[17] P. Taylor, *Text-to-Speech Synthesis*, Cambridge University Press, 2009.

[18] M. Lane, "Mobile Technology for Learning: Benefits and Challenges," *Educational Technology Research and Development*, vol. 69, no. 4, pp. 1895–1910, 2021.

[19] "Script Rehearser - Helping actors rehearse their lines," *ScriptRehearser.com*, 2024. [Online]. Available: https://www.scriptrehearser.com/#about [Accessed on 28-Apr-2025].
[20] "Rehearsal® Pro — The App For Actors," *RehearsalPro.com*, 2024. [Online]. Available: https://rehearsal.pro/
[21] "Rehearsal Pro – The App for Actors," RehearsalPro.com, 2020. [Online]. Available: https://rehearsal.pro/ [Accessed: 28-Apr-2025]
[22] "LineLearner — Memorize Lines for Plays, Film and TV," *LineLearnerApp.com*, 2024. [Online]. Available: https://apps.apple.com/us/app/linelearner/id368070258 [Accessed: 28-Apr-2025]
[23] "Scene Partner – The Rehearsal App," MyTheaterApps.com, 2020. [Online]. Available: https://scenepartner.ai. [Accessed: 28-Apr-2025]
[24] "ColdRead Mobile App," ColdRead App, 2021. [Online]. Available: https://coldreadapp.com/ [Accessed: 28-Apr-2025].
[25] Google Cloud, "Speech-to-Text Documentation," 2024. [Online]. Available: https://cloud.google.com/speech-to-text [Accessed: 28-Apr-2025].
[26] Expo Documentation, "Speech Recognition API," 2025. [Online]. Available: https://docs.expo.dev/versions/latest/sdk/speech/ [Accessed: 28-Apr-2025].
[27] Apple Developer, "Speech Framework," 2024. [Online]. Available: https://developer.apple.com/documentation/speech [Accessed: 28-Apr-2025].
[28] S. Deerwester et al., "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391-407, 1990.
[29] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1966.
[30] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of EMNLP-IJCNLP*, 2019.
[31] C. K. Lo and K. Hew, "The impact of web-based collaborative learning on students' academic achievement: A meta-analysis," *Educational Research Review*, vol. 24, pp. 52–68, 2018.
[32] D. McTear, "The Rise of Conversational Interfaces: A New Kid on the Block?," in *Proceedings of the International Workshop on Multimodal Interaction*, 2017.
[33] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive Science*, vol. 12, no. 2, pp. 257-285, 1988.
[34] Expo Documentation, "AsyncStorage API," 2025. [Online]. Available: https://docs.expo.dev/versions/latest/sdk/async-storage/ [Accessed: 28-Apr-2025].
[35] M. Offline, "Building Offline-First Applications," 2023. [Online]. Available: https://offlinefirst.org/ [Accessed: 28-Apr-2025].
[36] J. Martin, "Designing Data-Intensive Applications," *O'Reilly Media*, 2017.
[37] Expo, "What is Expo?", Expo, 2025. [Online]. Available: https://expo.dev. [Accessed: 28-Apr-2025].
[38] React Native, "Learn React Native", React Native, 2025. [Online]. Available: https://reactnative.dev. [Accessed: 28-Apr-2025].

[39] FastAPI, "FastAPI Documentation", FastAPI, 2025. [Online]. Available: https://fastapi.tiangolo.com. [Accessed: 28-Apr-2025].
[40] SQLite, "SQLite Documentation", SQLite, 2025. [Online]. Available: https://www.sqlite.org. [Accessed: 28-Apr-2025].
[41] Expo, "Expo Speech Recognition Documentation", Expo, 2025. [Online]. Available: https://docs.expo.dev. [Accessed: 28-Apr-2025].
[42] Google Cloud, "Cloud Speech-to-Text API", Google Cloud, 2025. [Online]. Available: https://cloud.google.com/speech-to-text. [Accessed: 28-Apr-2025].
[43] Hugging Face, "Sentence Transformers for Semantic Search", Hugging Face, 2025. [Online]. Available: https://huggingface.co/sentence-transformers. [Accessed: 28-Apr-2025].
[44] Google AI, "Universal Sentence Encoder", TensorFlow, 2025. [Online]. Available: https://www.tensorflow.org/hub. [Accessed: 28-Apr-2025].
[45] AsyncStorage documentation. "React Native AsyncStorage." [Online]. Available: https://react-native-async-storage.github.io/async-storage/ [Accessed: 28-Apr-2025].