UNIVERSITY OF LEICESTER

School of Computing and Mathematical Sciences

CO7201 Individual Project

Preliminary 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: 3299
28th February 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 February 2025

Contents

1. Aims and Objectives:

Aim:

Learning lines is a fundamental challenge for actors, as they often rely on partners or colleagues to rehearse, receiving feedback manually. This project aims to develop an AI-powered rehearsal assistant that enables actors to practice their lines independently by providing real-time feedback, error detection, and interactive scene partner simulations using text-to-speech (TTS) and speech recognition technologies.

Motivation:

Actors must master their lines with precision, but traditional rote memorisation is often ineffective. The most obvious method is repeated repetition [2]. Research suggests that simple repetition, known as maintenance rehearsal, does not significantly enhance long-term retention [1]. Instead, actors rely on elaborative rehearsal, a method that emphasises understanding the meaning of the material, forming connections with prior knowledge, and immersing themselves in their character's perspective. While rehearsing with a partner can improve recall and delivery, finding a reliable rehearsal companion is not always feasible. Some actors turn to self-rehearsal, but without external feedback, they may struggle to identify mistakes or refine their performance

The objective of this project aims to bridge these challenges by offering an AI-powered rehearsal assistant that simulates an interactive scene partner. By integrating advanced text-to-speech (TTS) and speech recognition technologies, the system will enable actors to rehearse anytime, receive real-time feedback, and refine their performances independently, making the rehearsal process more accessible, efficient, and engaging.

Main Challenges:

1. **Speech Recognition Accuracy** – Identifying spoken lines accurately while accounting for variations in phrasing and meaning without mistakenly marking correct lines as errors.
2. **Natural-Sounding Text-to-Speech** – Developing distinct character voices that sound realistic and engaging rather than robotic.
3. **User Experience & Accessibility** – Creating an intuitive interface that is easy to use for actors of all levels, ensuring seamless script uploading and rehearsal interactions.
4. **Real-Time Feedback Mechanism** – Implementing a system that detects mistakes and provides corrective suggestions instantly without disrupting the actor's flow.
5. **Customization & Personalization** – Allowing actors to adjust voice settings, track their progress, and receive tailored recommendations for improvement.
6. **Offline and Online Functionality** – Ensuring the app remains functional even without an internet connection while offering cloud-based enhancements when online.

2. Requirements

Essential Features:

- **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:** Analyze 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.

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.

Optional Enhancements:

- **Collaborative Rehearsal**: Support multi-user rehearsal, allowing multiple actors to practice scenes together.
- **Performance Reports**: Generate rehearsal history and performance analytics for users.
- **Dark Mode**: Provide a dark theme for eye comfort during extended practice sessions.
- **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.
- **OCR for Script Digitization**: Allow users to take a picture of a printed script and convert it into digital text for easy management.
- **Cloud-Based Online Version**: Develop an enhanced online version with advanced AI models and cloud-based backend infrastructure.

3. Technical Specification

Building an AI-powered rehearsal assistant requires selecting the right tools and technologies based on cost, efficiency, and scalability. Below is a detailed explanation of each technology used in the project

SNo.	Component	Technology	Free to use	Reason for using
1	Frontend	React Native (or Android studio)	Yes	Supports Cross-platform, open-source, Near-native performance, Single codebase
2	Backend	Node.js , Express.js, (Fast API, Flask python if needed)	yes	Lightweight and fast, simplify development as both frontend and backend are in Javascript
3	Database(offline)	SQLite	Yes	Local storage for offline functionality, Lightweight
4	Database(online)	PostgreSQL	Limited	Offers high Consistency, SQL based queries.
5	Speech Recognition	Vosk (offline), Whisper (online) (Google Cloud ASR)	Yes Yes Limited	Vosk – optimized for mobile, consumes less space, works offline Whisper- Open AI based high performance ASR
6	Text-to-Speech	Whisper, eSpeak (offline),	Yes	High quality, Provides Customizable voice
7	Text-to-Speech(Online)	Google TTS	Limited	Limited usage, High quality, Provides Customizable voice
8	NLP- Processing	Natural.js, Compromise.js	yes	To Identify paraphrases and similar meaning sentences, as simple string comparison is not effective
9	Cloud Deployment	Heroku	Yes basic	Quick and easy deployment of Node.js backend, Auto-Scaling available
10	Authentication	Firebase	Limited	To store user data, and Login authentication.

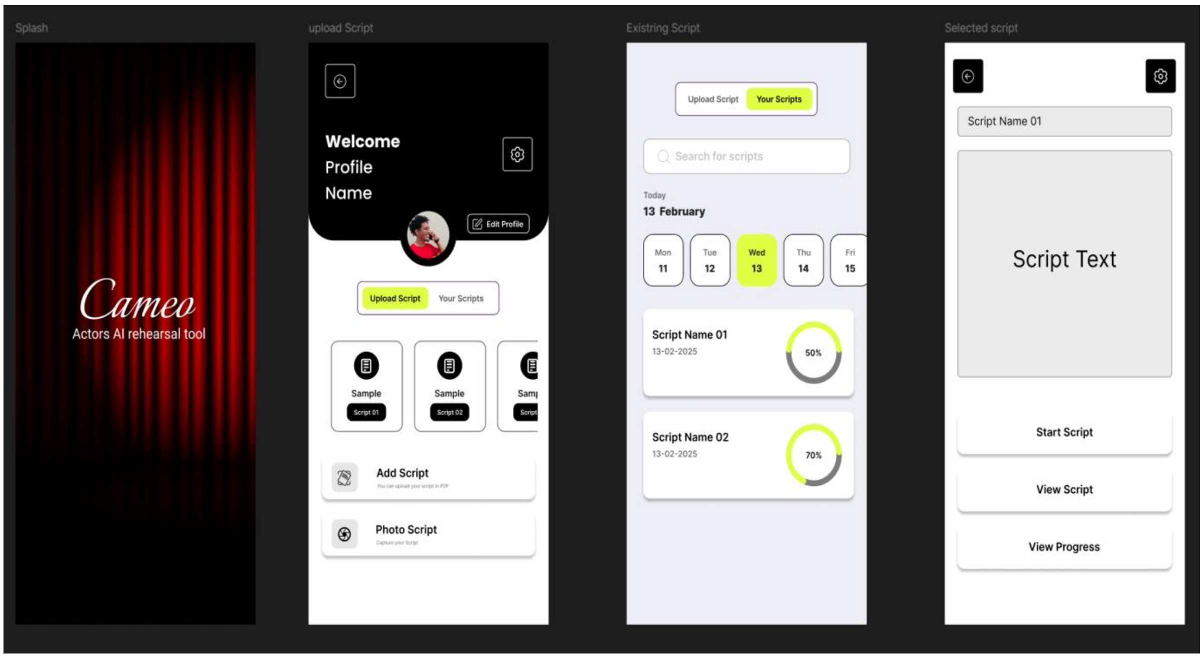
Architectural Overview

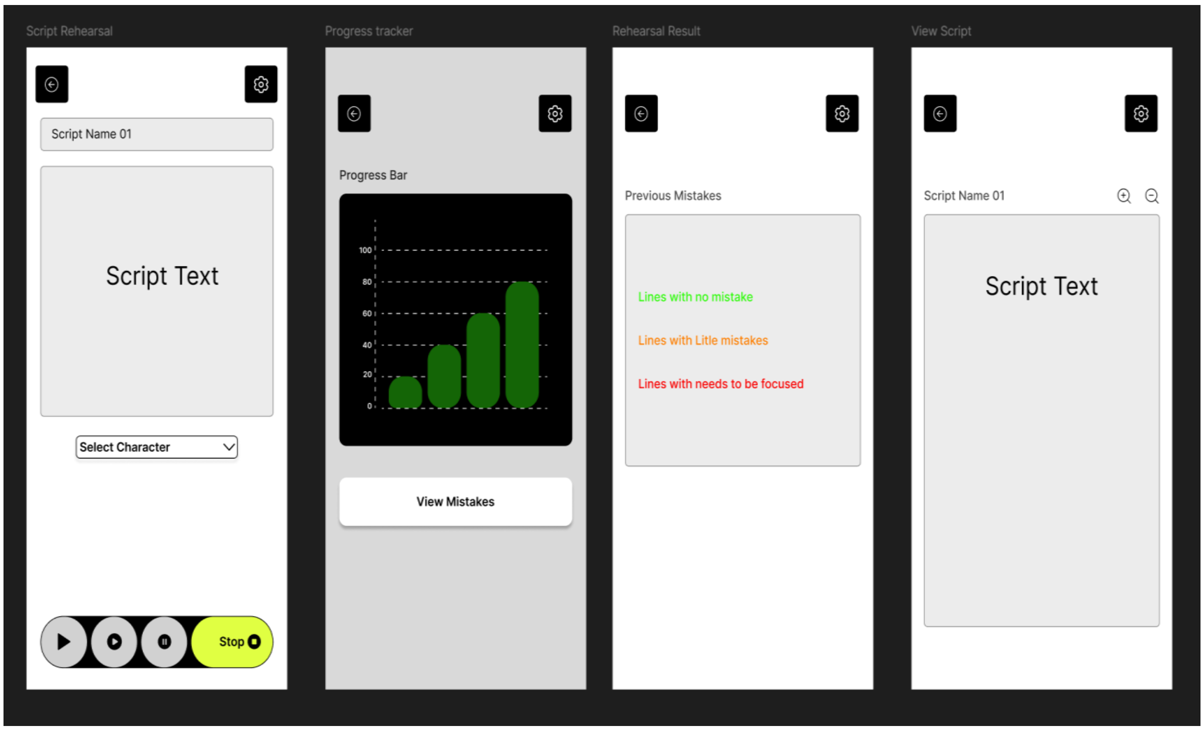
The application consists of the following key layers:

1. **Frontend (Mobile App) – React Native**
    a. Provides an intuitive UI for script management and rehearsal.
    b. Connects to local storage (SQLite) for offline functionality.
    c. Sends API requests to the backend for cloud synchronization (Online).
2. **Backend (Node.js with Express.js (or Flask, Fast API))**
    a. Handles authentication, script storage, and NLP processing for online mode.
    b. Manages cloud database (PostgreSQL or MongoDB).
    c. Provides API endpoints for data synchronization.

3. **Database Layer**
    a. Offline Mode: Uses SQLite to store scripts locally.
    b. Online Mode: Uses PostgreSQL or MongoDB for cloud-based storage.
    c. Automatic sync mechanism updates local scripts when online.
4. **AI Processing Layer**
    a. Speech Recognition:
        i. Uses Vosk API for offline speech to text conversion.
        ii. Uses OpenAI Whisper or Google Cloud ASR when online for better accuracy.
    b. Text-to-Speech (TTS):
        i. Uses Whisperrn or eSpeak for offline speech synthesis.
        ii. Uses Google TTS API when online for better speech quality.
    c. Error Detection:
        i. Utilizes NLP libraries like natural.js and compromise.js for sentence similarity matching.
        ii. Use Hybrid model if needed (Sentence-BERT, Levenshtein Distance, Jaccard Similarity)
5. **Cloud Services**
    a. Heroku for backend hosting.
    b. Firebase for user authentication and cloud script storage.

The Wireframe below is a conceptual User Interface for the Mobile app.





The Frontend is a mobile application developed using React Native with Expo as it allows to build a cross-platform mobile app with single codebase.

The Primary Screens needed for the app are:

SNo.	Screen Title	Purpose	Features	Implementation
1	Splash Screen	Display app name	Authentication if needed	Use React Native Splash screen API and Firebase google authentication
2	Upload Script	Allow users to upload or capture the script using the camera	Upload Script, List of recently opened scripts	Use Expo DocumentPicker for script upload and Expo camera to upload script
3	Uploaded Script	Display all uploaded scripts	Search for uploaded scripts, delete script, select or view script	Use React Native FlatList for displaying existing scripts.
4	Selected Script	Shows the selected script with various options	User can start the rehearsal, view the full script and view progress reports	Use React Navigation and on click buttons.
5	Script Rehearsal	The main rehearsal screen where users	Allow users to play full scripts,	Use React Native Audio for playback , Use Expo

		practice their lines	Start or stop rehearsal , select character and change AI voice.	speech or Google TTS API for text to speech and Whisper API for real time speech recognition or any other speech recognition model.
6	Progress Tracker	Tracks user performance and analytics	View rehearsal progress (accuracy vs attempts), Button to navigate to view previous attempt mistakes.	Use Recharts or victory,js for analytics visualization, Store progress in the database
7	Previous Result	Show past attempt mistakes in colour coded format	The correct sentences are in green, partially correct once are in orange and wrong sentence are in red colour.	Use React Native Text Styling
8	View Script	Allow users to read the full script	Users can Adjust Font size.	Use React Native Scroll View for scrolling through the script.
9	Settings	Allow users to edit their profile details	Users can change their profile name , profile picture , select dark mode	Save the details in Local storage (SQLite)

4. Requirements Evaluation Plan

Requirements Gathering:

1. **Consulting with the Assigned Professor**: Ensure that the project aligns with academic expectations and technical feasibility. Present the initial project idea and core functionalities. Discuss technical challenges, implementation constraints, and expected outcomes. Receive feedback on project scope, research methodologies, and evaluation strategies.
2. **Consulting with Potential Users**: Gather input from the potential users like actors, drama students etc.
3. **Market Research**: Identify existing rehearsal tools and their limitations to refine the application. Document strengths and weaknesses of each tool. Identify missing

functionalities that can be added or optimized. Few examples , Script Rehearser , Voice Dream (IOS), Speechmatics etc.

Development Phase Evaluation:

The development phase evaluation ensures that the AI-powered rehearsal assistant is built correctly, integrating all components seamlessly and efficiently. This phase involves **continuous testing, iterative improvements, and feedback loops**. We will implement an agile approach with multiple iterations, ensuring that features are tested and refined incrementally.

Frontend Testing : Ensure proper rendering and responsiveness of the user interface components or screens. Smooth transition between screens.

Backend Testing : Ensure the authentication works as expected with firebase. Validate and hit the endpoints for script upload , storage and retrieval works as expected. Perform Load test, stress test by simulating multiple users accessing the backend simultaneously and measure the responses.

AI model Testing : Asses the speech recognition accuracy across accents and speeds. In Text-to-speech, Check for character distinction, ensure users can adjust pitch, speed and gender of the voice . Test how well the system highlights and detects the mistake. Make sure the semantic understanding is maintained.

Feedback Iteration : After receiving necessary feedback from the Assigned Professor and targeted user optimize or proceed with the changes where ever needed.

Final Evaluation Phase

Validate that the system meets all essential requirements before deployment by performing end to end testing and usability testing.

SNo.	Requirement	Evaluation Method	Success Metric
1.	Mobile Application	Usability testing, UI/UX evaluation	Based on user satisfaction
2.	Text-to-Speech (TTS) and Customizable Voice Settings	Voice clarity tests, user feedback, performance benchmarking	More than 80% user should find voice differentiation
3.	Speech Recognition	Testing with different accents and speeds	Greater than 90% in speech to text conversion
4.	Semantic Understanding	NLP evaluation, user testing for paraphrased lines	More than 85% correct interpretation of minor variations
5.	Script Management System	Functionality testing	Successful script upload/edit/delete actions with no failures
6.	Deadline & Notifications	Functionality testing and frequency of notifications	90% or more timely notifications received by users

7.	Video Recording	Playback quality testing, storage efficiency test	Clear video output, no lag or major frame drops
8.	OCR for Script Digitization	Image-to-text accuracy testing	successful conversions with minimal errors
9.	Full-Script Playback	Audio playback quality test	clarity in AI-generated speech playback
10	Progress Tracker	Performance tracking evaluation	The insights generated has greater than 90% accuracy.
11	Cloud-Based Online Version	Cloud performance testing, load testing, synchronization validation	95% or more uptime, successful data sync across devices

5. Background Research and Reading list

Academic Research Paper:

1. "**NLP-Theatre: Employing Speech Recognition Technologies for Improving Accessibility and Augmenting the Theatrical Experience**" [3]:
   This paper introduces a system that streamlines the initiation of events to boost accessibility and enrich live theatrical performances. It employs Automatic Speech Recognition (ASR) to capture audio from live performances and match it with existing scripts, enabling real-time subtitle modifications and other synchronised activities.

2. "**Collaborative Storytelling with Human Actors and AI Narrators**"[4]:
   This research paper investigates the role of large language models, particularly GPT-3, as collaborative narrators in storytelling. The AI system works alongside human performers by monitoring plot advancement and character evolution, aiding in live shows.

3. "**A Study on Improving the Accuracy and Effectiveness of Similarity Detection Processes in Text Files Using NLP Techniques**"[5] : This paper explores different NLP techniques to identify the similarity in a sentence and a method to increase accuracy and efficiency.

4. "**Measurement of Text Similarity: A Survey**"[19]: This paper provides insights about various sentence similarity methods and compares it's advantages and disadvantages.

5. “**Is Noise Reduction Improving Open-Source ASR Transcription Engines Quality**”[21]. This paper explores different ASR model like Whisper and Vosk’s performance variations due to noise.

6. “**Mobile-based Speech Recognition for Early Reading Assistant**”[20]: The authors have developed a speech recognition based mobile app to help people learn English. The system has similar features to our project like Optical character recognition and Text to speech.

Reading list:

Frontend Development:

React Native: React Native is a cross-platform framework that enables frontend development for both iOS and Android with a single codebase [6]. The documentation provides essential guidelines for UI development, navigation, and performance optimization. Expo simplifies the React Native development as it has pre-configured libraries [7].

Backend Development:

Node.js and Express.js is used for backend development. The documentation provides the required insights about the framework like API request and routing etc[8][9]. Firebase Docs helps with user authentications [10]. Heroku documentation will guide us on deploying and managing backend efficiently [11].

Database Management:

SQLite and PostgreSQL are used for storing the progress reports, user data and scripts offline and online respectively [12][13].

AI Models and Processing:

Speech Recognition we can use Vosk API for offline speech recognition and Whisper ASR or Google Speech-to-text API documentation can be used to understand its working to tune the model as per out need[14][15].
Text-to-Speech processing Whisper Github Documentation is useful for offline processing and Google’s Text-to-speech cloud based API documentation can be referred[16].
Natural.js and Compromise.js Documentation can help us to understand NLP tasks like synonyms detection and semantic understanding[17][18],

6. Time-plan and Risk Plan

Time Plan:

17th Feb – 3rd March: Finding project scope, objectives, feasibility. Gather the required Reading list and Academic Papers. Preliminary report writing and submission. Meeting with assigned Professor and getting feedback about project descriptions and scope. Gathering requirements and features needed for the project. Planning the tasks and fixing on the technical specifications like the tech stack, AI models and cloud services.

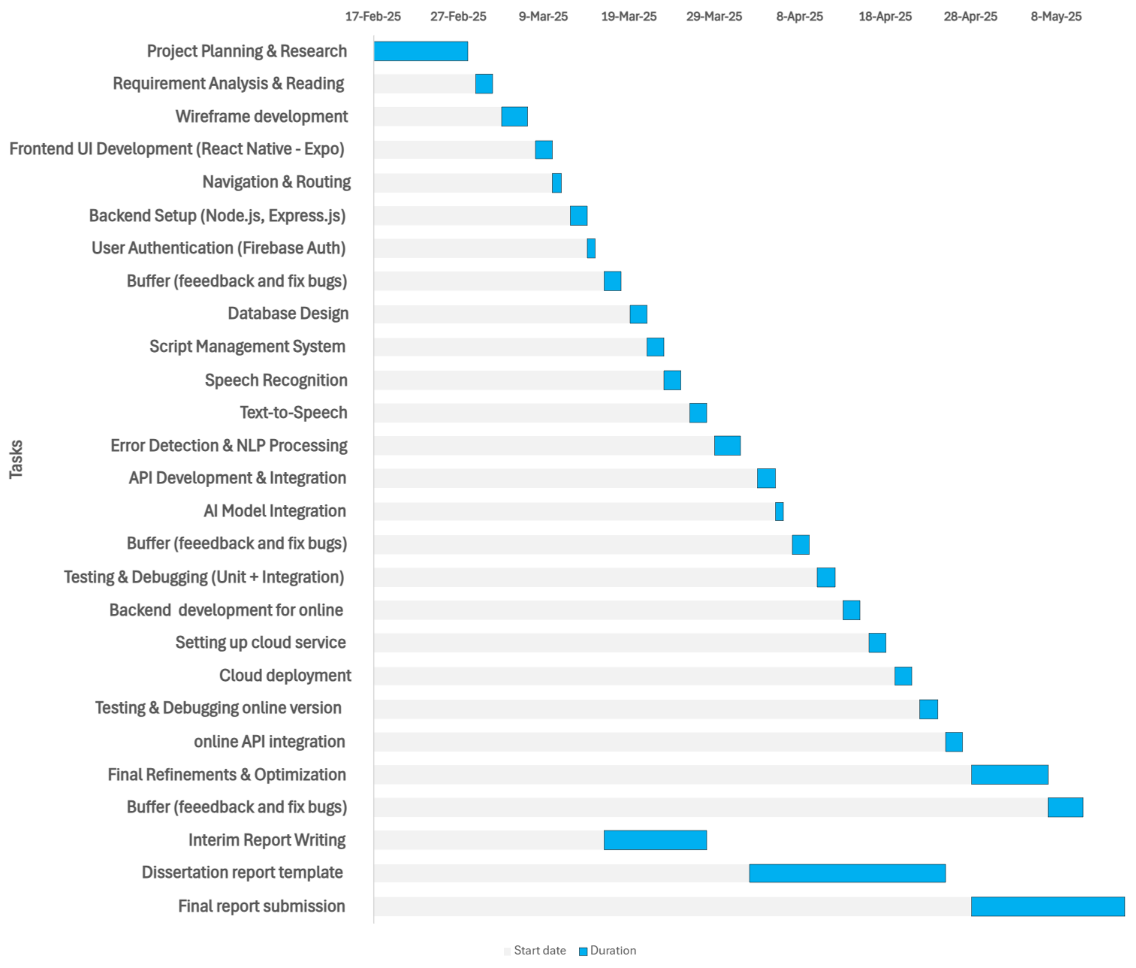
4th March – 11th March: Create visual designs and layout structures for the app. Deciding on the number of screens and flow of the app. Design the final user interface in Figma and get feedback from the targeted user and supervisor. Implement the UI using react native.

12th March – 23rd March: Set up the backend with necessary API routes and middleware. Implement login and authentication security. Fix bugs and refine based on early testing and feedback on the Frontend. Define schema for scripts, progress tracking, and user settings. Develop script upload, storage, edit, and delete features.

24th March – 9th April : Implement and test speech-to-text accuracy across different accents. Implement and test character voice differentiation and customization. Ensure semantic understanding, synonym detection, and real-time feedback. Interim Report submission. Connect frontend with backend for smooth data interaction. Address issues found during development and API testing.

10th April – 27th April: Develop backend support for cloud-based features. Configure cloud storage and backend deployment settings. Deploy the backend and ensure all cloud services function correctly. Test cloud services, database connections, and app stability. Integrate online AI models and APIs with the cloud-based system. Dissertation template report submission.
28th April – 16th May: Optimize app performance, UI improvements, and AI model adjustments. Fix last-minute issues based on feedback before submission. Submit the final dissertation report.

Gantt chart:



Risk Plan:

Any delays or failures could impact the overall progress thus below is the risk assessment plan which outlines the potential risks and alternative solution

SNo.	Risk	Impact	Mitigation
1	Speech Recognition Errors or Low accuracy	Medium	Use multiple APIs (Vosk, Whisper ASR) for comparison. Fine-tune NLP models for better accuracy. Provide manual correction options.
2	UI Complexity	Low	Conduct iterative user testing to refine usability.
3	API Integration Issues	Medium	Use offline models (Vosk, espeak, whisper-rn) when API services fail.
4	Backend Failure	Low	Use the offline mode
5	Offline AI and NLP model accuracy is low	High	Implement multiple AI models and use online based models to improve accuracy.

6	Time Constraints	Medium	Prioritize core functionalities like Speech Recognition, TTS and Error Detection
7	Offline AI model size is too high	Medium	Move to online based model
8	AI model library for Next.js is not accurate, lags or consumes more memory	High	Build an offline version and build a python based backend like Fast API or Flask as python has a vast AI library support

Continuous monitoring and implementing the necessary control measures can ensure a successful project execution within the given timeframe.

7. References

[1] How Actors Remember Their Lines, The MIT press reader website accessed 28 February 2025. https://thereader.mitpress.mit.edu/how-actors-remember-their-lines/

[2] How Do Actors Memorize Lines? Here's How 9 Professionals Do It, Backstage website, accessed 28 February 2025,
https://www.backstage.com/magazine/article/backstage-experts-answer-ways-quickly-memorize-lines-6719/

[3] Alkiviadis Katsalis, Konstantinos Christantonis, Charalampos Tsioustas, "NLP-Theatre: Employing Speech Recognition Technologies for Improving Accessibility and Augmenting the Theatrical Experience" Intelligent Systems and Applications, September 2022

[4] Boyd Branch, Piotr Mirowski, Kory W. Mathewson, "Collaborative Storytelling with Human Actors and AI Narrators" arXiv preprint September 2021.

[5] Noor Abdulmuttaleb Jaafar, "A Study on Improving the Accuracy and Effectiveness of Similarity Detection Processes in Text Files Using NLP Techniques" Al-Esraa University College Journal for Engineering Sciences September 2024

[6] React Native, React Native Documentation website, accessed 28 February 2025. https://reactnative.dev/docs/getting-started

[7] Expo, Expo Documentation website, accessed 28 February 2025. https://docs.expo.dev/

[8] Node.js, Node.js Documentation website accessed 28 February 2025.
https://nodejs.org/docs/latest/api/

[9] Express.js, Express.js Documentation website accessed 28 February 2025.
https://devdocs.io/express/

[10] Firebase, Firebase Documentation website accessed 28 February 2025. https://firebase.google.com/docs/auth

[11] Heroku, Heroku Documentation website accessed 28 February 2025. https://devcenter.heroku.com/categories/reference

[12] SQLite, SQLite Documentation website accessed 28 February 2025. https://www.sqlite.org/docs.html

[13] PostgreSQL, PostgreSQL Documentation website accessed 28 February 2025. https://www.postgresql.org/docs/

[14] Vosk API, Vosk API Documentation website accessed 28 February 2025. https://alphacephei.com/vosk/

[15] Google Speech to text (STT) API, Google STT API Documentation website accessed 28 February 2025. https://cloud.google.com/speech-to-text/docs

[16] Whisper, Whisper github Documentation website accessed 28 February 2025. https://github.com/mybigday/whisper.rn

[17] Natural.js, Natural.js npm Documentation website accessed 28 February 2025. https://www.npmjs.com/package/natural/v/1.0.1

[18] Compromise.js, Compromise.js npm Documentation website accessed 28 February 2025. https://www.npmjs.com/package/compromise

[19] Jiapeng Wang and Yihong Dong , "Measurement of Text Similarity: A Survey" MDPI journal August 2020

[20] Muhammad Adlan Mohd Faisol, Siti Azura Ramlan, Aini Hafizah Mohd Saod, Aiza Mahyuni Mozi and Fazrul Faiz Zakaria "Mobile-based Speech Recognition for Early Reading Assistant" J.Phys 2021

[21] Asma Trabelsi, Laurent Werey, S´ ebastien Warichet and Emmanuel Helbert,"Is Noise Reduction Improving Open-Source ASR Transcription Engines Quality" ICAART 2024.