Cairo University

Faculty of Computers and Artificial Intelligence

Computer Science Department

المَقْرَأة

وَرَتِّلِ الْقُرْآنَ تَرْتِيلًا

Supervised by:

Dr. Hanaa Bayoumi

TA. Amany Hesham

Implemented by:

20210614	Ahmed Mohamed Sallam
20201126	Omar Walid Ahmed
20201146	Mohamed Gamal Abdelaziz
20200378	Farah Tawfik Salem
20200436	Mohamed Gaber Mohamed

Graduation project

Academic Year 2023-2024

Final Discussion

Table of Contents

List of Figures

List of Tables

Chapter 1 : Introduction

1.1. Motivation:

The goal of Al-Maqraa is to help Muslims overcome two significant obstacles: pronouncing the Quran correctly and memorizing it. Because of scheduling conflicts and distance limitations, traditional techniques, which sometimes require a human instructor, are not as accessible. By providing a technical solution that is accessible around-the-clock, Al-Maqraa seeks to close this gap by enabling users to practice and get feedback whenever it is convenient for them.

commitment to making Quranic learning accessible to Muslims globally. In a world marked by diverse time zones, cultural contexts, and accessibility challenges, the platform seeks to break down barriers and provide a universally available solution.

Al-Maqraa has its roots in the numerous, varied Muslim communities that exist in the USA, Canada, and the EU. Only 1400 mosques are accessible for the 25,770,000 Muslims in the EU, which restricts access to Quranic education. There are just 200 mosques in Canada, although there are 33,000,000 Muslims, and 2481 mosques in the USA, where there are 3.45 million Muslims.

Introducing new technology using Artificial Intelligence in the field of the Arabic language and the Holy Quran.

1.2. Problem definition

Fixed timings for sheik instruction are a common feature of traditional Quranic learning approaches. The lack of a resource that is accessible around-the-clock makes it difficult for people to study the Quran at their convenience.

The absence of immediate feedback on individual Quranic recitation impedes the learning process. Without timely and constructive feedback, individuals may struggle to identify and rectify errors in their pronunciation and memorization.

Regretfully, all the systems available for memorizing the Qur'an simply check if the

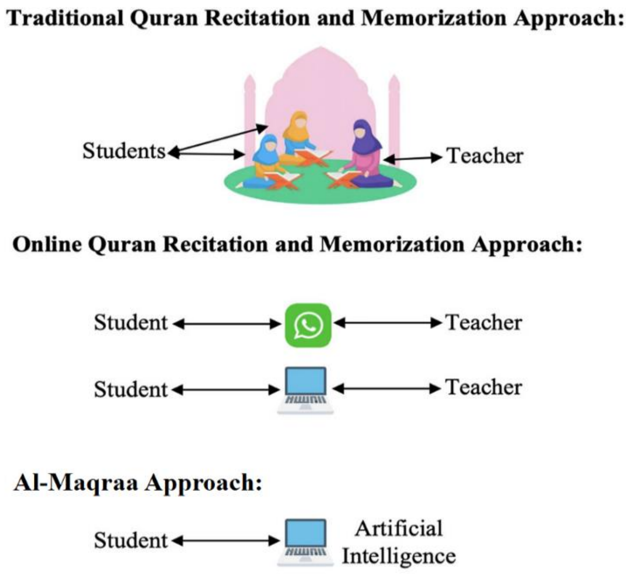

Traditional Quran Recitation and Memorization Approach:
Students
Teacher
Online Quran Recitation and Memorization Approach:
Student
Teacher
Student
Teacher
Al-Maqraa Approach:
Student
Artificial Intelligence

Fig. 1 Traditional and Al-Maqraa Quran recitation Approach

text has been committed to memory; they do not check to see if the pronunciation is correct.

1.3. Objective:

The primary objective of this research is to develop an innovative mobile application named Al-Maqraa, designed to assist Muslims in accurately reciting and memorizing the Quran. This application addresses the limitations of traditional Quranic learning methods, such as the reliance on human instructors and fixed schedules, by leveraging advanced technologies to provide a comprehensive, accessible, and flexible solution. Al-Maqraa proposes a mobile application that features an embedded machine learning (ML) model to evaluate users' Quranic recitations. The application employs two sophisticated ML models, CNN-CTC (Convolutional Neural Network with Connectionist Temporal Classification) and RNN-CTC (Recurrent Neural Network with Connectionist Temporal Classification), to ensure a thorough and precise assessment of users' recitations. This dual-model approach guarantees a detailed examination, offering users immediate and constructive feedback on their pronunciation and memorization, thereby significantly enhancing their learning experience. Developed using Flutter, the application offers a seamless and user-friendly interface that is compatible with multiple platforms, including iOS and Android. The backend, built using the Microsoft .NET Framework and PostgreSQL, ensures robustness, reliability, and scalability, capable of handling a growing number of users and large datasets without compromising performance. Additionally, the application includes features for tracking memorization progress, bookmarking verses, and daily activity tracking, alongside access to recordings of renowned Imams for guidance. Security and reliability are prioritized, with robust measures to protect user data and maintain minimal downtime. By integrating these features, Al-Maqraa aims to deliver a modern, technology-driven solution for Quranic learning that is accessible, flexible, and effective for Muslims worldwide. This research contributes to the field of Quranic education by integrating advanced ML models to provide an innovative approach to learning and recitation

1.4. Gantt Chart

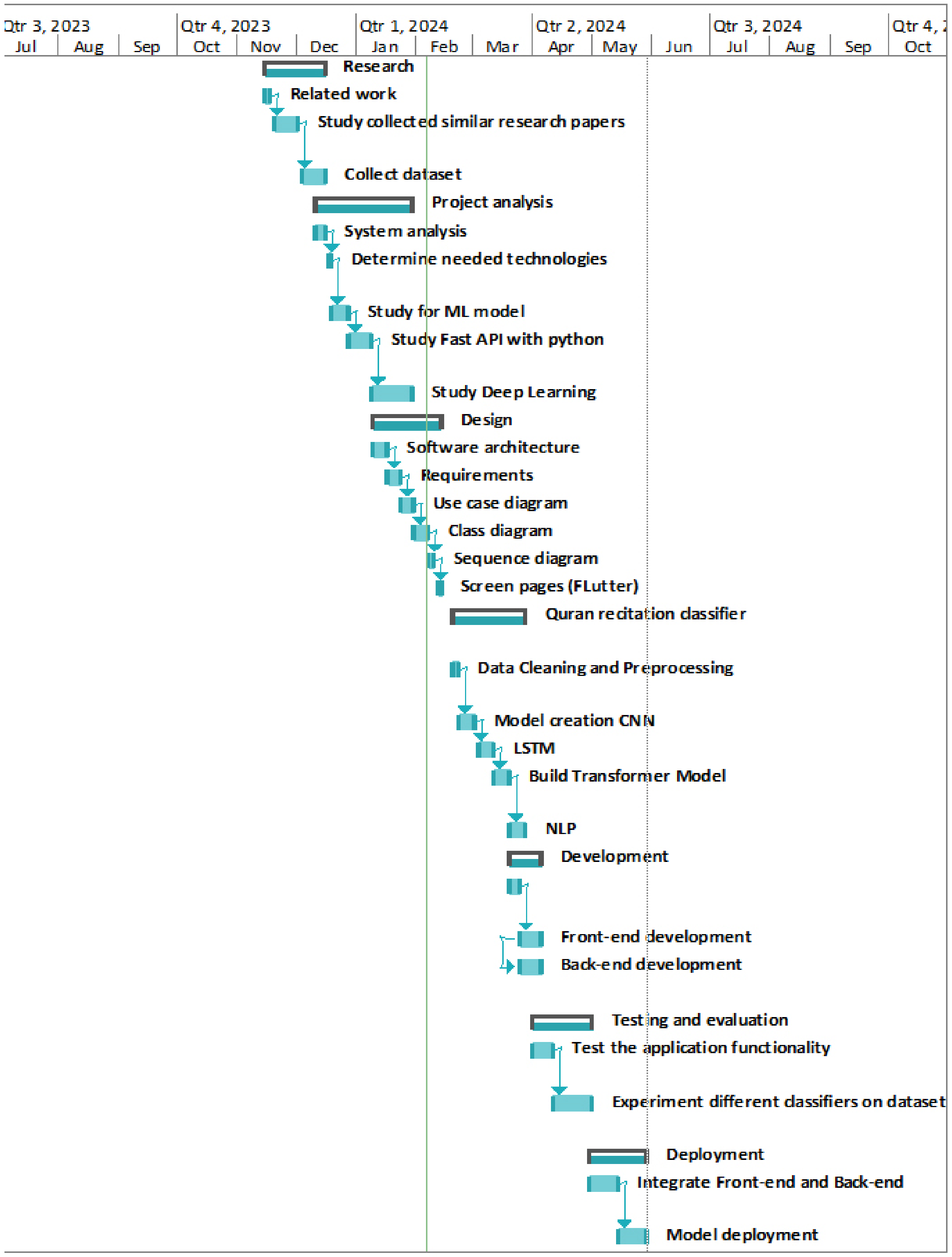


Fig. 2 - Gantt Chart

1.5. Project development methodology

The development of the Al-Maqraa application will follow an Agile methodology to ensure flexibility, continuous feedback, and iterative improvement throughout the project lifecycle. The Agile approach will facilitate collaboration among the development team, stakeholders, and end-users, ensuring the delivery of a high-quality product that meets the users' needs. Initially, the project will involve planning and requirement analysis, where the primary objectives, scope, and deliverables will be established. Detailed requirement analysis sessions with stakeholders will be conducted to gather functional and non-functional requirements, and features will be prioritized. In the design phase, a tiered system architecture will be developed, including presentation, business, and data layers. Wireframes and prototypes will be created using tools like Figma to ensure an intuitive and user-friendly interface, while technical specifications for frontend, backend, ML models, and database structures will be outlined. During the development phase, the project will be divided into multiple sprints, each lasting 2-4 weeks, with specific goals and deliverables. Flutter will be used for frontend development to create cross-platform user interfaces for iOS and Android. Backend services will be implemented using Microsoft .NET Framework and PostgreSQL for database management. CNN-CTC and RNN-CTC models will be developed and trained using PyTorch and TensorFlow, and Flask will be utilized for deploying the machine learning models. Integration will ensure seamless communication between frontend, backend, and ML models. The application will be deployed on Azure to leverage its robust cloud services. Maintenance and iteration will involve addressing any bugs reported by users, continuously gathering user feedback to implement new features or improvements, and regularly updating the application to ensure compatibility with new operating system versions and technologies. This comprehensive methodology aims to deliver a modern, high-quality application for Quranic learning.

1.6. Main Techniques, Technologies, and Application

1.6.1. Fronted Technologies

1. Flutter:

an open-source framework that Google developed. Utilizing the same code, it is used to create apps for several platforms. Consequently, we could utilize it to create an application with the same code that runs on both iOS and Android. It may also be applied to the construction of user interfaces for websites.

Application: Mobile frontend.

1.6.2. Backend Technologies

1. Microsoft .NET Framework:

.NET Framework is a robust and versatile software development framework developed by Microsoft. It provides a comprehensive programming model, a vast class library, and runtime support for building and deploying various types of applications. Originally released in the early 2000s, .NET Framework has since evolved into a family of frameworks and technologies that cater to different application scenarios.

Application: Backend of the system

2. Flask

fast web framework for building APIs with Python, ideal for deploying machine learning models. It provides high performance, automatic generation of interactive API documentation, and easy integration with machine learning libraries. FastAPI supports asynchronous programming, making it efficient for handling concurrent requests. It is widely used for creating scalable and production-ready ML model deployment services.

Application : Deploy Machine Learning Models.

3. PostgreSQL

It's an object-relational database management system that's available for free. It supports user-defined types and application features like XML and JSON. It is safe, dependable, and quick for both read and write operations, making it appropriate for big systems or systems with big datasets.

Application: querying and managing databases

1.6.3. ML Technologies

1. Deep Learning

It is a multi-layered kind of neural network and machine learning. It may operate under supervision or independently and mimics the functioning of the human brain. But instead, we'll employ a supervised method with a labeled dataset.

Application: Virtual teacher for the Quran

2. PyTorch

PyTorch is an open-source machine learning framework developed by Facebook. It provides a flexible and dynamic computational graph, making it well-suited for deep learning tasks. PyTorch is widely used for building and training neural networks, and it has gained popularity in the research community due to its intuitive and Pythonic syntax. The framework supports dynamic computation, allowing for easy experimentation and model development.

Application: Development of CNN – CTC Architecture.

3. TensorFlow

TensorFlow is an open-source machine learning framework developed by Google that facilitates building and training neural networks. It supports various platforms, including CPUs, GPUs, and TPUs, enabling scalable deployment across devices. TensorFlow offers flexible tools and libraries for research and production, with high-level APIs like Keras for easy model creation. It is widely used in academia and industry for tasks ranging from natural language processing to computer vision.

Application: Development of RNN - CTC Architecture.

1.6.4. Cloud Computing Tools

1. Bitvise SSH Client:

a powerful Windows-based tool for secure remote access to servers via SSH. It provides a user-friendly interface for terminal access, file transfer, and tunneling. The client supports advanced features like port forwarding, command scripting, and strong encryption to ensure secure communications. It is widely used for system administration, remote support, and secure file management.

Application: Using to access Bibliotheca Alexandrina - Cloud Computing

2. Bash (Bourne Again Shell)

a Unix shell and command language, widely used for scripting and automation on Unix-like operating systems. It provides a command-line interface for interacting with the system, executing commands, and running scripts. Bash is powerful for automating repetitive tasks, managing system processes, and handling complex workflows. It is a default shell on many Linux distributions and macOS, making it essential for system administrators and developers.

Application: Using to optimize GPU operations

1.7. Report Organization

The report is organized into several chapters, each detailing different aspects of the Al-Maqraa project, from related work to system design, implementation, and testing.

Chapter 2: Related Work This chapter reviews the closest existing examples related to the Al-Maqraa project. It compares these examples with Al-Maqraa, highlighting the main differences and unique features of our project. References to previous works and projects are provided to support the comparisons and discussions.

Chapter 3: System Analysis

3.1 Project Specification

3.1.1 Functional Requirements: This section lists the essential functionalities that the Al-Maqraa application must provide, such as user authentication, Quran recitation, memorization tracking, and feedback mechanisms.

3.1.2 Non-functional Requirements: This section outlines the performance, scalability, security, and usability criteria that the application must meet to ensure a smooth user experience and robust operation.

3.2 Use Case Diagrams

System Component Diagram: Illustrates the overall system architecture and the interaction between different components.

System Class Diagrams: Provides a detailed view of the classes and their relationships within the system.

Sequence Diagrams: Depicts the flow of operations and interactions between system components for various use cases.

Project ERD (Entity Relationship Diagram): Shows the data model and the relationships between different data entities.

System GUI Design: Describes the graphical user interface layout and design elements to ensure an intuitive user experience.

Chapter 4: System Design This chapter delves into the detailed design of the Al-Maqraa system. It covers the design principles and patterns used, the architecture of the system, and how the components are integrated. It includes detailed diagrams and descriptions of the system's structure and design logic.

Chapter 5: Implementation and Testing This chapter discuss the implementation phase of the Al-Maqraa project, detailing the development process, the technologies used, and the steps taken to build the application. It also covers the testing strategies and methodologies employed to ensure the application meets all specified requirements. This includes unit testing, integration testing, user acceptance testing, and performance testing to validate the functionality, reliability, and efficiency of the system.

Chapter 2: Related Work

2.1. Commercial applications

❖ There are multiple websites with similar features as

1. <u>Tarteel.ai [1]:</u> Using deep neural networks, this speech recognition-based Quran application facilitates reading and memorization of the Holy Quran. The program offers a memorizing mode where the user may record his memory of the Quran while the verses are concealed. Words will emerge when they are correctly spoken by the user. On the other hand, if a word is mispronounced by the user, it will be highlighted in red. Additionally, the program has audio search features for both verses and chapters. You must purchase the Premium edition of the program to access all its features. The application concentrates on accurately spoken words without considering Tajweed rules and word diacritical. This edition has a monthly fee of $7.50.

2. <u>Tamkeen [2]:</u> A Quran application that uses speech recognition technology to facilitate reading and memorization of the Holy Quran. In the memorizing mode, the verses are buried until the user successfully recites them. Until the user corrects his recital, a mispronounced verse will remain hidden. The program prioritizes accurately spoken words above Tajweed rules and word diacritical marks.

3. <u>Tasmee [3]:</u> A Quran application that uses speech recognition technology to facilitate reading and memorization of the Holy Quran. AI-based personal recitation is one of the services it offers. Additionally, the program offers a memorizing mode in which all of the verses disappear unless they are said properly by the user. Sharing one's recitation with others is another feature that it offers. Because the program takes into account Tajweed restrictions, it might not be appropriate for novice users.

❖ **Main difference between Al-Maqraa and the other application:**

• Al-Maqraa offers free access to its extensive Quranic learning platform, in contrast to many of its rivals who charge a monthly fee.

• Al-Maqraa does an excellent job of supporting Arabic diacritical marks, which helps students pronounce and recite words correctly. This feature improves the learning process overall and distinguishes it from rivals.

• Al-Maqraa goes beyond simple error identification to focus on both obvious and concealed faults. this ensures that the student Pronounced correctly.

• Al-Maqraa facilitates the seamless connecting of verses, hence enhancing the accuracy and fluidity of recitation. With a correct comprehension of the verse sequence, this function improves the user's capacity to memorize and recite the Quran.

• Memorization progress monitoring, session history, and streaks are all available in Al-Maqraa. These tools provide students the ability to track their progress over time, encouraging continuous involvement and development.

Features	**Al-Maqraa**	**Tarteel.ai**	**Tamkeen**	**Tasmee**
Supported platform	Android and IOS	Android and IOS	Android	Android and IOS
Price	Free	$7.50 per Month	Free	Free
Support diacritics?	Yes	No	No	No
Connect the verse with the next?	Yes	No	Yes	Yes
Type of error the app focuses on	Obvious errors and Hidden errors	-	Obvious errors	Hidden errors
Streaks	Yes	Yes	No	No
Session history	Yes	Yes	No	No
Memorization progress	Yes	Yes	No	No
Limitations		Does not consider obvious errors, such as missing the first letter of a verse Does not ensure that the learner reads the verses in the correct order	• Encounters some speech recognition errors	• Encounters some speech recognition errors • Sometime doesn't Recognize the correct word

Table 1 - Comparison between Al-Maqraa and the Other applications

2.2. Existing Literature Reviews

Table 2 - Topics covered in previous review papers.

Subject	Basic Information		Recitation recognition	Recitation verification	Traditional speech recognition methods	End-to-end speech recognition methods	Audio feature extraction techniques	Datasets	Mobile applications	Traditional Models	End-to-end Models
	Authors	Year									
Quranic Verse Recitation Recognition Module for Support in j-QAF Learning: A Review [4]	Zaidi Razak.et al	2008	✓	✗	✓	✗	✓	✗	✗	ANN HMM VQ	
Improve design for automated Tajweed checking rules engine of qur'anic verse recitation [5]	Noor Jamaliah & M. Y. Zulkifli &Zaidi Razak	2011	✓	✓	✓	✗	✓	✓	✗	HMM MLLR	
Quranic Verses Verification using Speech Recognition Techniques [6]	Mohammed.et al	2015	✓	✓	✓	✗	✓	✗	✗	ANN HMM VQ	
Automatic speech recognition for the holy Qur'an, a review [7]	Bilal Yousfi1 & Akram M Zeki	2016	✓	✓	✓	✗	✓	✗	✗	MFCC HMM	
Verification system for Quran recitation recordings [8]	Sherif Mahdi Abdo & Ayat Hafzalla	2017	✓	✓	✓	✗	✓	✗	✗	ANN HMM VQ	
Deep Diacritics-Based Recognition Model for Arabic Speech[9]	Sarah S. Alrumiah	2023	✓	✗	✓	✓	✓	✓	✗		TDNN-CTC RNN – CTC Transformers
Intelligent Quran Recitation Recognition and Verification: Research Trends and Open Issues [10]	Sarah S. Alrumiah	2023	✓	✓	✓	✓	✓	✓	✓	HMM GMM VQ K-Means SVM TDNN	LSTM MaLSTM HMM+BLSTM

As shown in Table 3, work from the review articles that have already been published on Quran recitation voice recognition and verification. The specifics of conventional voice recognition techniques, which rely on linguistic and acoustic models as well as pronunciation dictionaries, were covered in the brief review studies that are now available [4–10]. The preprocessing and feature extraction approaches were also covered in those works. The first review paper on Quran recitation recognition was released in 2008 and included the first attempts at recitation recognition using the Sphinx toolkit [4]. Furthermore, a revised version of [5] including Quran recitation verification efforts was published in 2011. Moreover, other review articles addressing voice recognition and verification efforts for traditional Quran recitation were released in 2015 [6], 2016 [7], and 2017 [8]. Verification methods have also been proposed in [6, 8].
The authors of [8] suggested a verification method to recognize the user and fix them when a recitation error happens. However, a verification method based on matching steps—that is, matching the uttered words with the target words until the recitation stops—was presented by the authors in [6].
Consequently, the current review papers have just covered the conventional speech recognition technique. On the other hand, this work addresses both end-to-end and conventional voice recognition techniques used for Quran verification and recitation. Additionally, while Quran learning mobile applications were covered in [9], the instructor's real-time audio-based recitation recognition and verification applications were not included. In order to highlight the existing shortcomings and restrictions in practical applications, these kinds of applications are demonstrated in this study.
Therefore, this study's distinctive contributions are as follows: • It covers all voice recognition and verification work related to Quran recitation from 2006 to 2021. Taking into account efforts relevant to both conventional and end-to-end voice recognition.
Talking about the limits of the real-time, instructor-free mobile applications for recitation identification and verification that are now available.

Chapter 3: System Analysis

3.1 Project specification

3.1.1 Functional requirement

User Authentication: Users will be able to create accounts and log in securely.

Quran Recitation: Users can recite a specific (Ayahs).

Quran Memorization:

Users can access a dedicated section for Quran memorization.
Include audio prompts for users to repeat and practice memorized verses.

Listening for Imam:

Provided for users to listen to recorded Quranic recitations by renowned Imams.
Allow users to choose their preferred Imam for recitations.

Bookmarking and Favorites: Users will be able to bookmark specific (Surahs).

Daily Tracker:

Implement a daily tracker to record and display the user's Quranic activities.
Provide visual representations of daily progress and achievements.

3.1.2 Non-functional Requirements:

Performance: Response time for user actions will be within acceptable limits.

Scalability: The system will handle a growing number of users and content without affecting performance.

Reliability: The application will be available 24/7 with minimal downtime.

Security: Protect user data from unauthorized access.

Compatibility: The application will be compatible with different versions of mobile operating systems and with various mobile devices and screen sizes.

Usability: The user interface will be intuitive and easy to navigate.

Provide accessibility features for users with disabilities.

Maintainability: Implement a maintainable code structure for easy updates and future enhancements.

3.2. Use Case Diagram

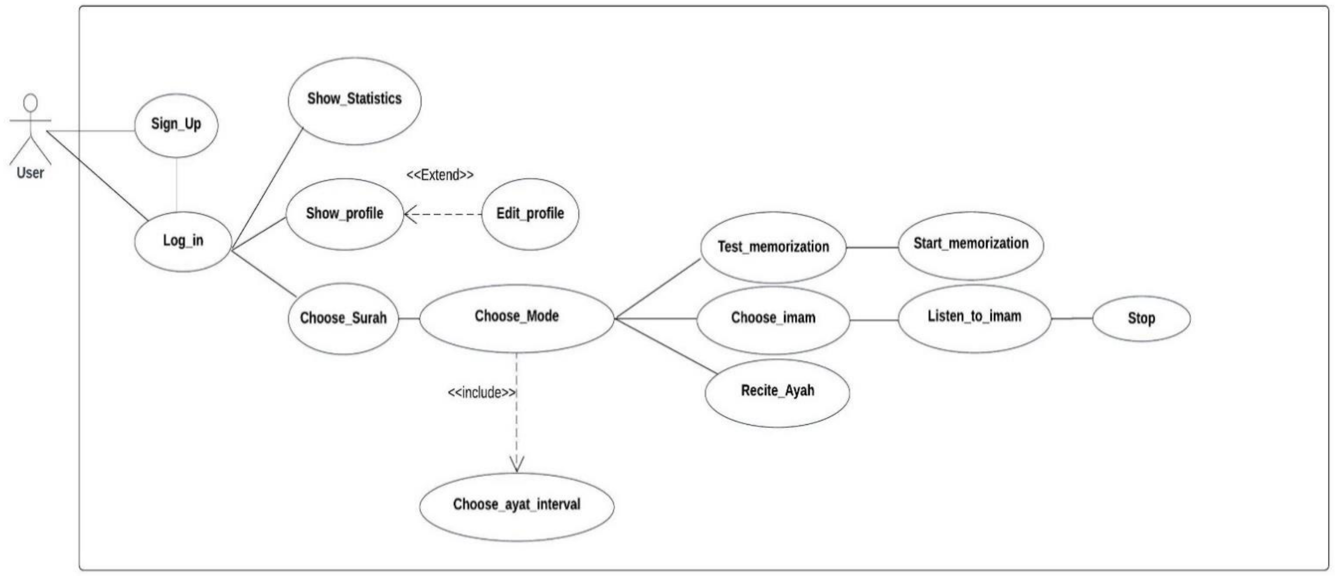


Fig. 3 - Use Case Digram

Chapter 4: System Design

4.1. System Architecture

• We'll employ a tiered architecture in which every layer communicates with every other layer.

❖ Presentation layer:

• In charge of overseeing user engagement with the system.

❖ Business Layer:

• Responds to requests from the display layer, applies logic to them, and retrieves data from the data layer.

❖Data Layer:

• In charge of keeping track of the data and establishing a connection to retrieve it from the back end.

• consists of a Redis database that handles caching and a local database.

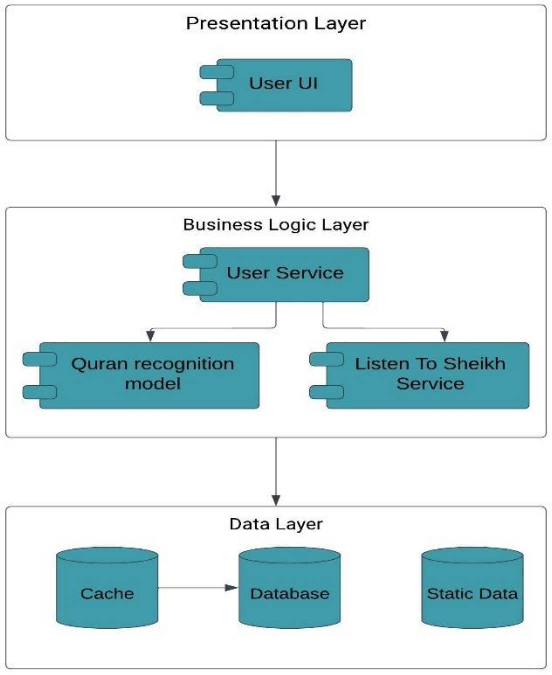


Fig. 4 System Architecture

4.2. System Component Diagram

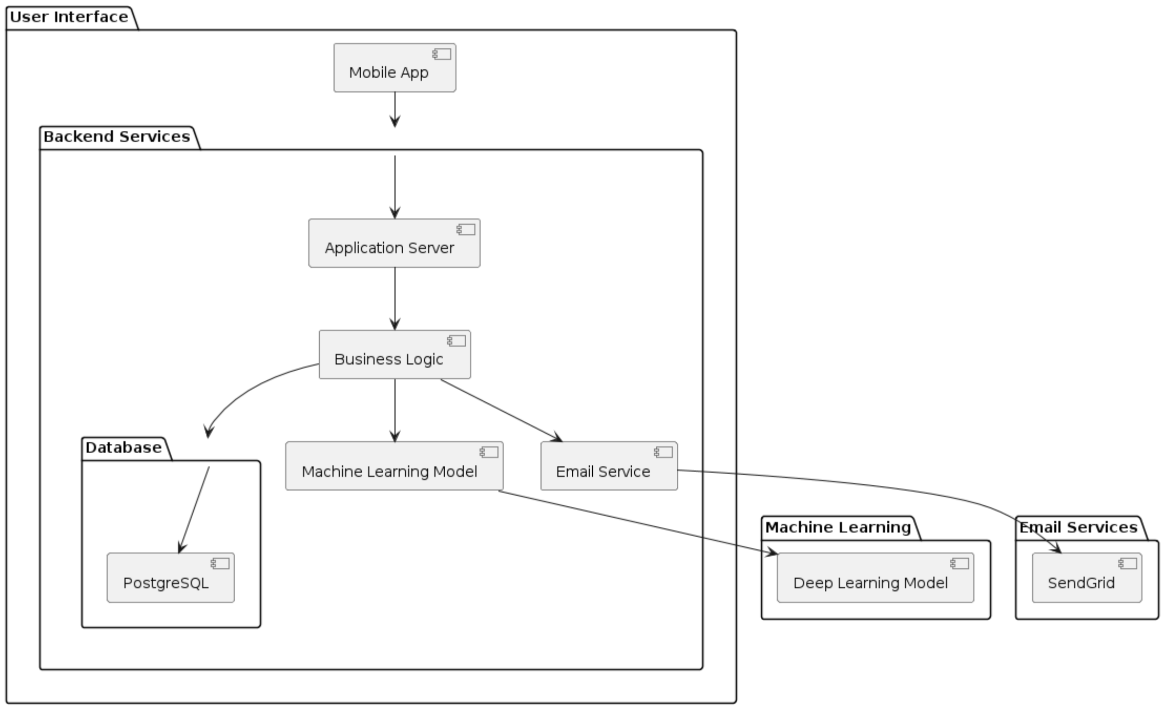


Fig . 5 *System Component Diagram*

4.3. Class Diagram

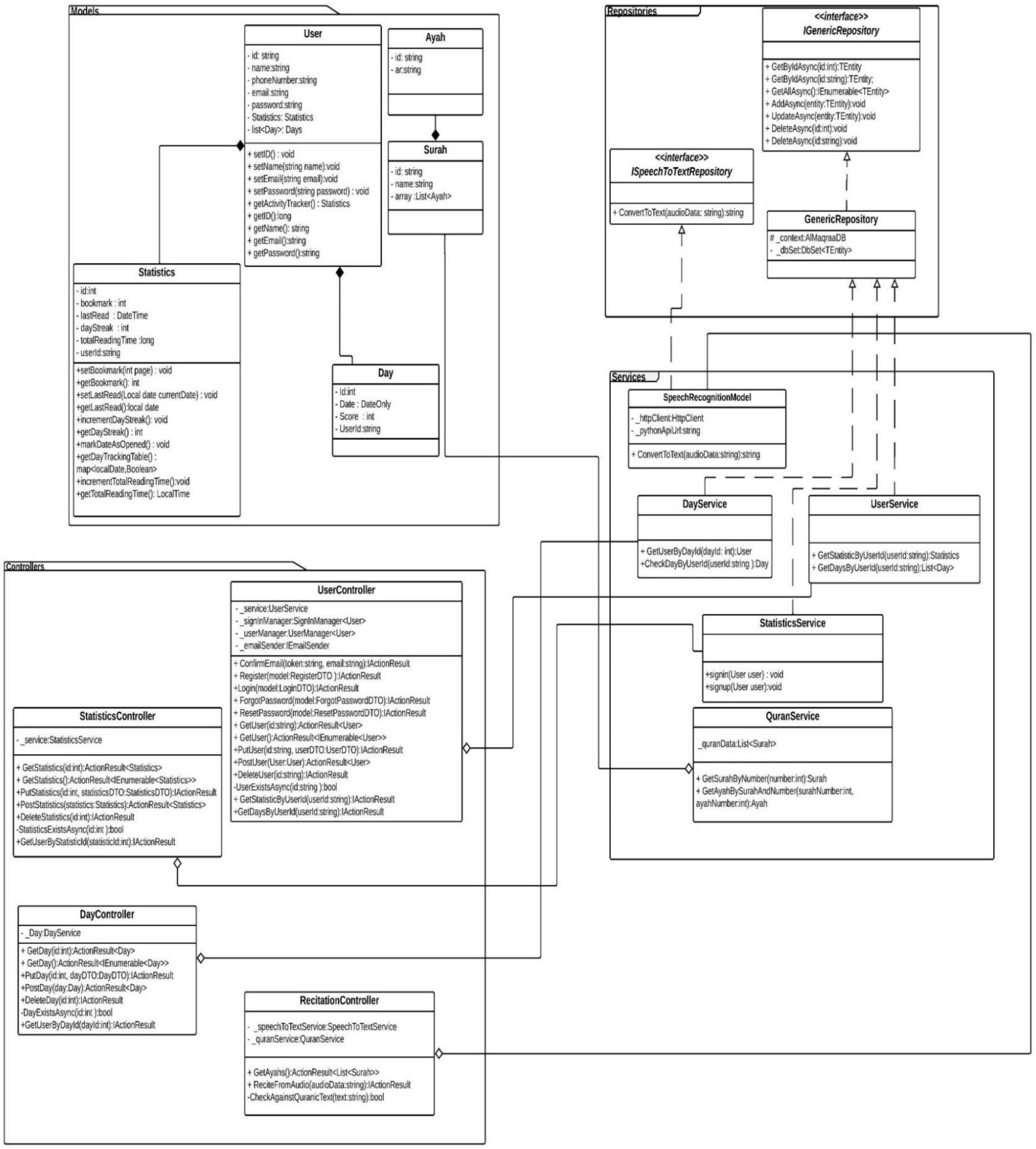

Models
User
- id: string
- name:string
- phoneNumber:string
- email:string
- password:string
- Statistics: Statistics
- list<Day>: Days
+ setID() : void
+ setName(string name):void
+ setEmail(string email):void
+ setPassword(string password) : void
+ getActivityTracker() : Statistics
+ getID():long
+ getName(): string
+ getEmail():string
+ getPassword():string
Ayah
- id: string
- ar:string
Surah
- id: string
- name:string
- array :List<Ayah>
Statistics
- id:int
- bookmark : int
- lastRead : DateTime
- dayStreak : int
- totalReadingTime :long
- userId:string
+setBookmark(int page) : void
+getBookmark(): int
+setLastRead(Local date currentDate) : void
+getLastRead():local date
+incrementDayStreak(): void
+getDayStreak() : int
+markDateAsOpened() : void
+getDayTrackingTable() : map<localDate,Boolean>
+incrementTotalReadingTime():void
+getTotalReadingTime(): LocalTime
Day
- Id:int
- Date : DateOnly
- Score : int
- UserId:string
Repositories
<<interface>>
IGenericRepository
+ GetByIdAsync(id:int):TEntity
+ GetByIdAsync(id:string):TEntity;
+ GetAllAsync():IEnumerable<TEntity>
+ AddAsync(entity:TEntity):void
+ UpdateAsync(entity:TEntity):void
+ DeleteAsync(id:int):void
+ DeleteAsync(id:string):void
<<interface>>
ISpeechToTextRepository
+ ConvertToText(audioData: string):string
GenericRepository
# _context:AlMaqraaDB
- _dbSet:DbSet<TEntity>
Services
SpeechRecognitionModel
- _httpClient:HttpClient
- _pythonApiUrl:string
+ ConvertToText(audioData:string):string
DayService
+ GetUserByDayId(dayId: int):User
+CheckDayByUserId(userId:string):Day
UserService
+ GetStatisticByUserId(userId:string):Statistics
+ GetDaysByUserId(userId:string):List<Day>
StatisticsService
+signin(User user) : void
+signup(User user):void
QuranService
_quranData:List<Surah>
+ GetSurahByNumber(number:int):Surah
+ GetAyahBySurahAndNumber(surahNumber:int, ayahNumber:int):Ayah
Controllers
UserController
- _service:UserService
- _signInManager:SignInManager<User>
- _userManager:UserManager<User>
- _emailSender:IEmailSender
+ ConfirmEmail(token:string, email:string):IActionResult
+ Register(model:RegisterDTO):IActionResult
+Login(model:LoginDTO):IActionResult
+ ForgotPassword(model:ForgotPasswordDTO):IActionResult
+ ResetPassword(model:ResetPasswordDTO):IActionResult
+ GetUser(id:string):ActionResult<User>
+ GetUser():ActionResult<IEnumerable<User>>
+PutUser(id:string, userDTO:UserDTO):IActionResult
+PostUser(User:User):ActionResult<User>
+DeleteUser(id:string):IActionResult
-UserExistsAsync(id:string):bool
+ GetStatisticByUserId(userId:string):IActionResult
+GetDaysByUserId(userId:string):IActionResult
StatisticsController
- _service:StatisticsService
+ GetStatistics(id:int):ActionResult<Statistics>
+ GetStatistics():ActionResult<IEnumerable<Statistics>>
+PutStatistics(id:int, statisticsDTO:StatisticsDTO):IActionResult
+PostStatistics(statistics:Statistics):ActionResult<Statistics>
+DeleteStatistics(id:int):IActionResult
-StatisticsExistsAsync(id:int):bool
+GetUserByStatisticId(statisticId:int):IActionResult
DayController
- _Day:DayService
+ GetDay(id:int):ActionResult<Day>
+ GetDay():ActionResult<IEnumerable<Day>>
+PutDay(id:int, dayDTO:DayDTO):IActionResult
+PostDay(day:Day):ActionResult<Day>
+DeleteDay(id:int):IActionResult
-DayExistsAsync(id:int):bool
+GetUserByDayId(dayId:int):IActionResult
RecitationController
- _speechToTextService:SpeechToTextService
- _quranService:QuranService
+ GetAyahs():ActionResult<List<Surah>>
+ ReciteFromAudio(audioData:string):IActionResult
-CheckAgainstQuranicText(text:string):bool

Fig . 6 *Class Digram*

4.4. Sequence Diagram

4.4.1. User Choose to listen to Quran.

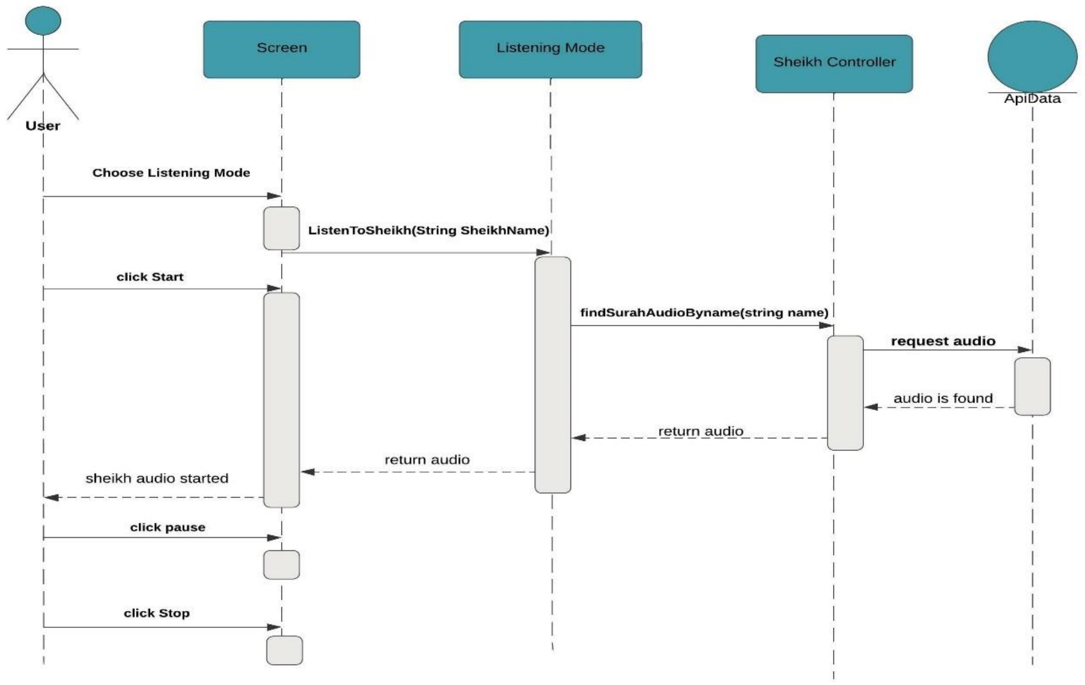


Fig . 7 User Chooses to listen to sheikh

4.4.2. User Choose to Recite to Quran.

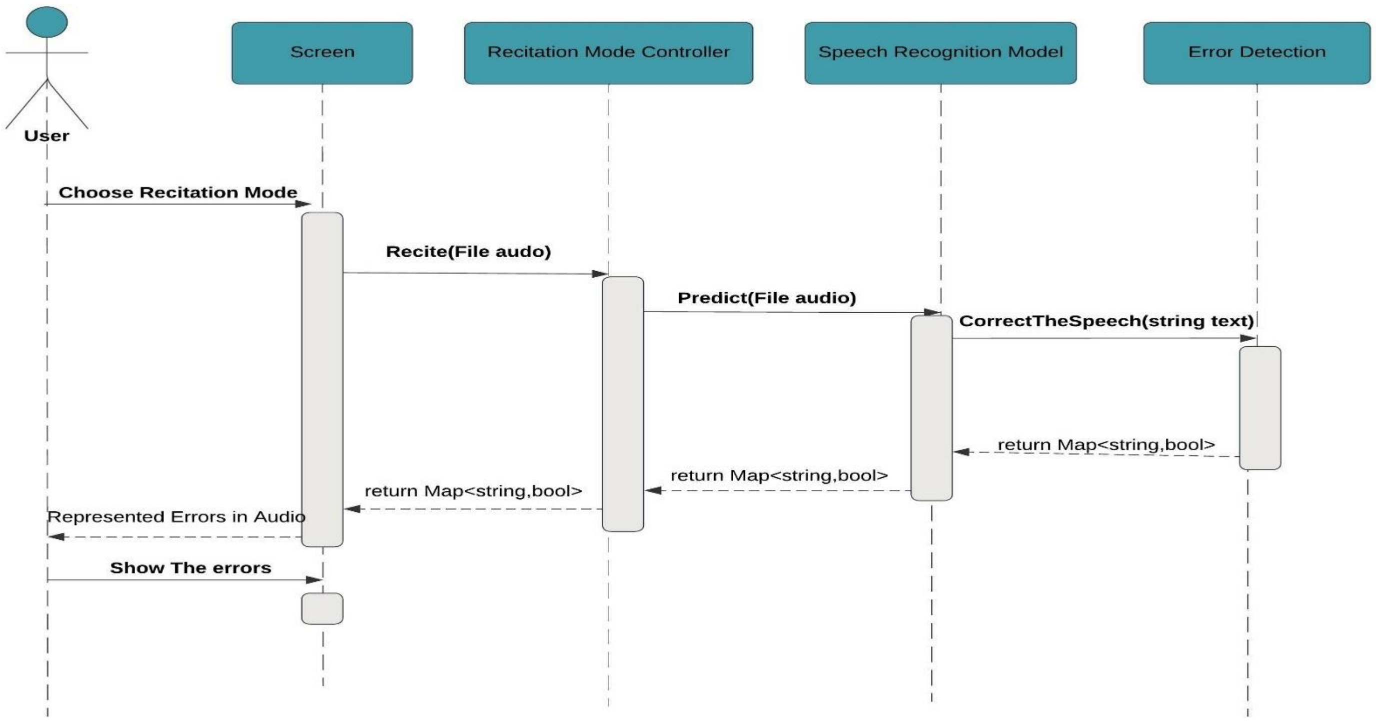


Fig . 8 - User Choose to Recite Quran

4.4.3. The user chooses the Quran memorization test.

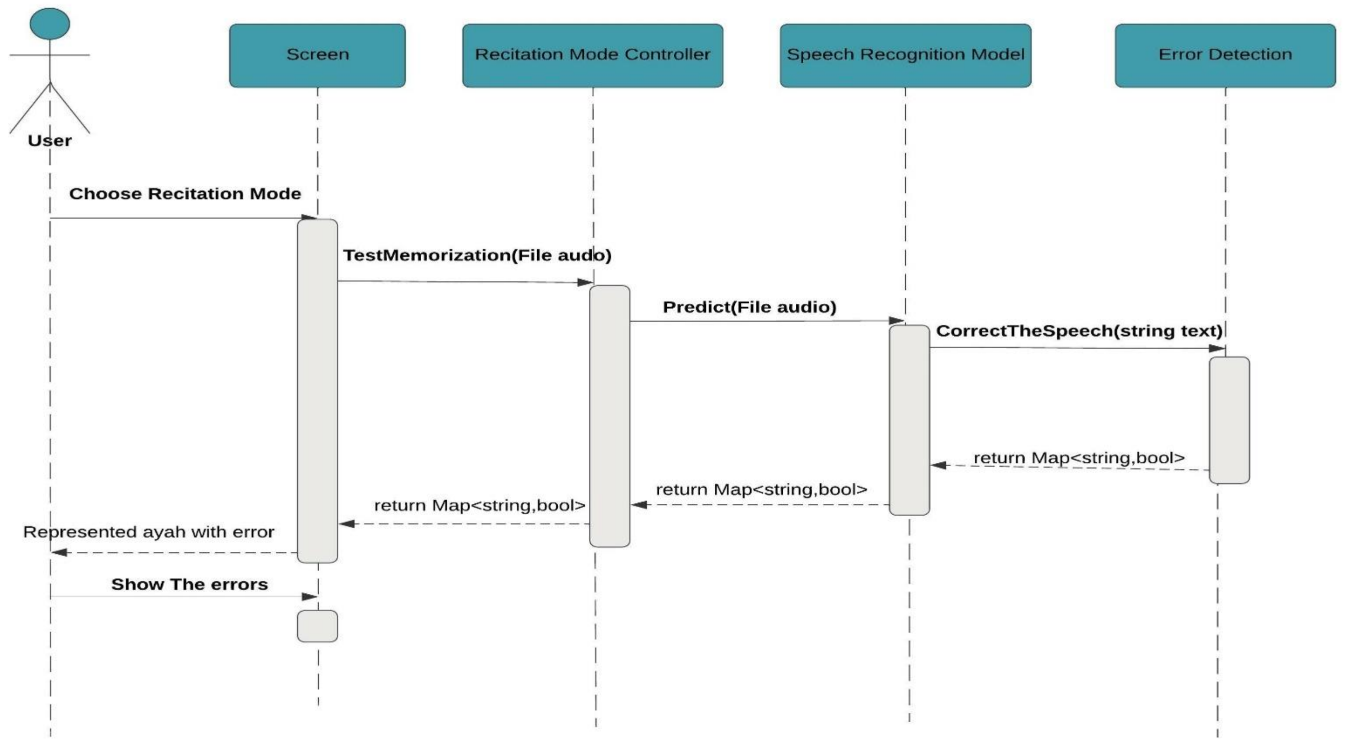


Fig . 9 Quran memorization test

4.5. Entity Relationship Diagram (ERD)

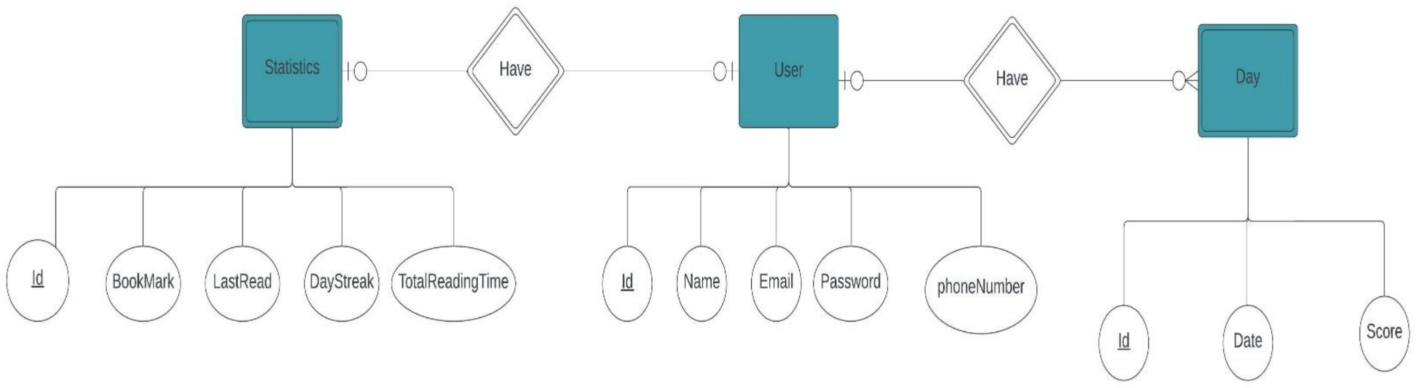
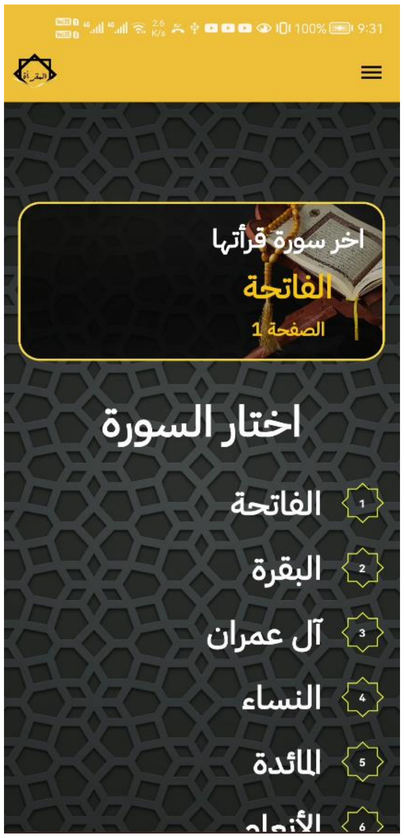
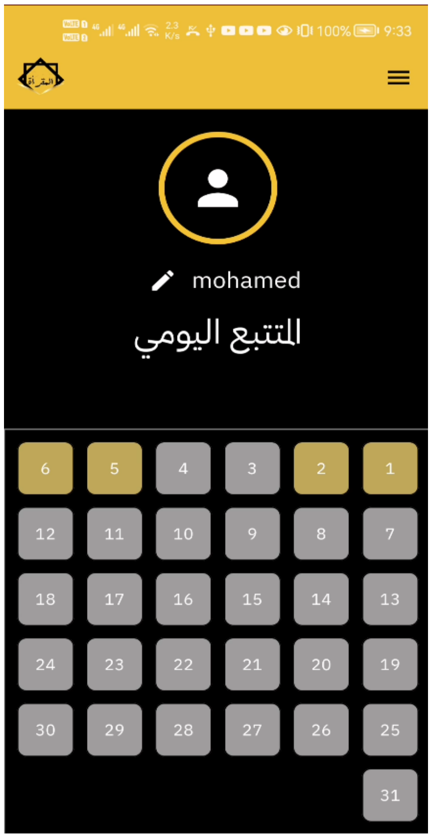


Fig . 10 ERD

4.6. System GUI Design





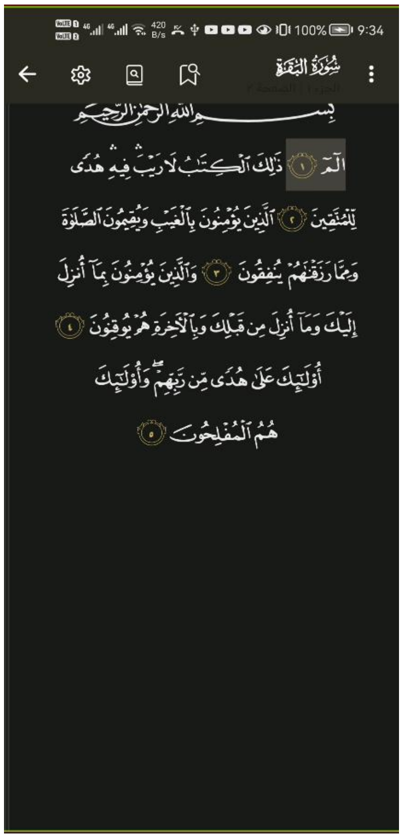
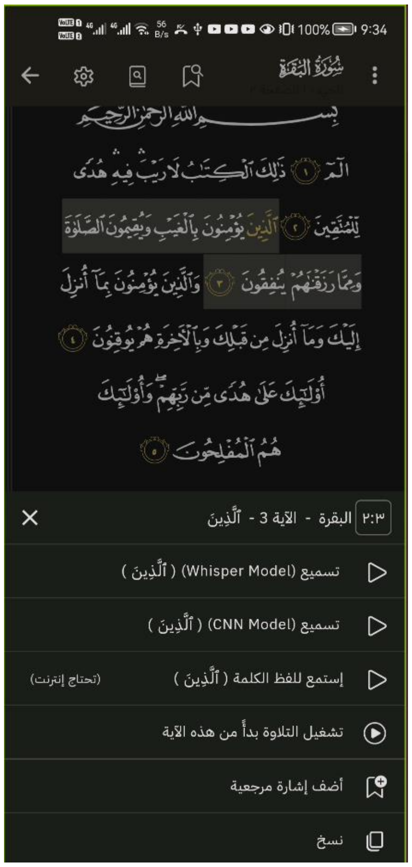
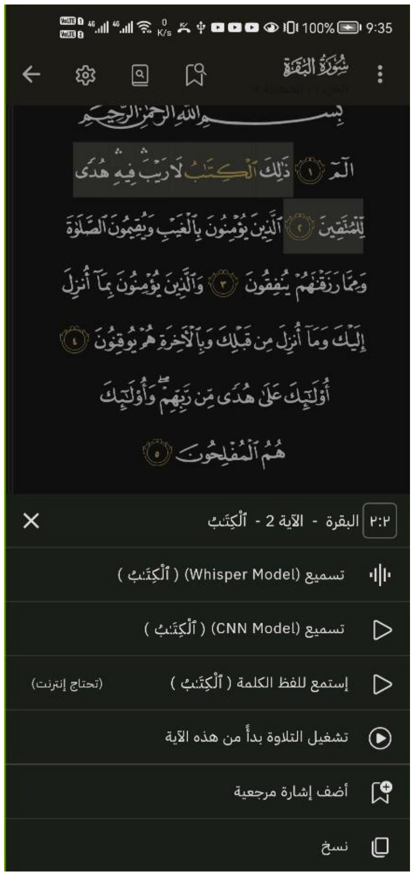


Fig . 11 Sample of System GUI



9:34
سُورَةُ البَقَرَةِ
٢:٣ البقرة - الآية 3 - ٱلَّذِينَ
تسميع (Whisper Model) (ٱلَّذِينَ)
تسميع (CNN Model) (ٱلَّذِينَ)
إستمع للفظ الكلمة (ٱلَّذِينَ) (تحتاج إنترنت)
تشغيل التلاوة بدأ من هذه الآية
أضف إشارة مرجعيه
نسخ



9:35
سُورَةُ البَقَرَةِ
٢:٢ البقرة - الآية 2 - ٱلْكِتَٰبُ
تسميع (Whisper Model) (ٱلْكِتَٰبُ)
تسميع (CNN Model) (ٱلْكِتَٰبُ)
إستمع للفظ الكلمة (ٱلْكِتَٰبُ) (تحتاج إنترنت)
تشغيل التلاوة بدأ من هذه الآية
أضف إشارة مرجعيه
نسخ

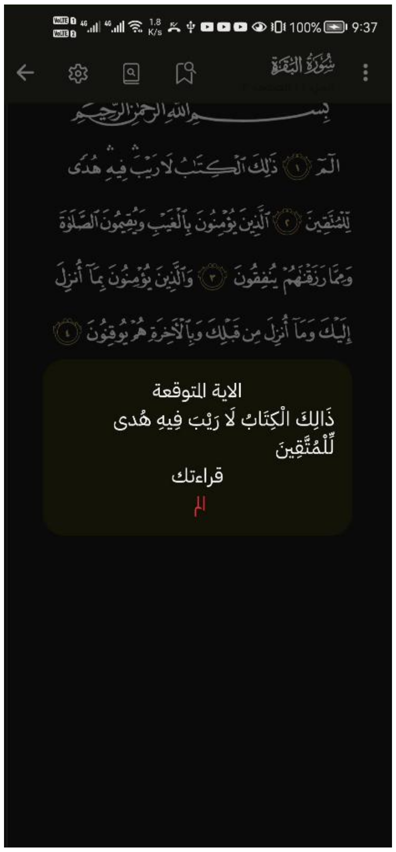


Fig . 12 Sample of System GUI - 2

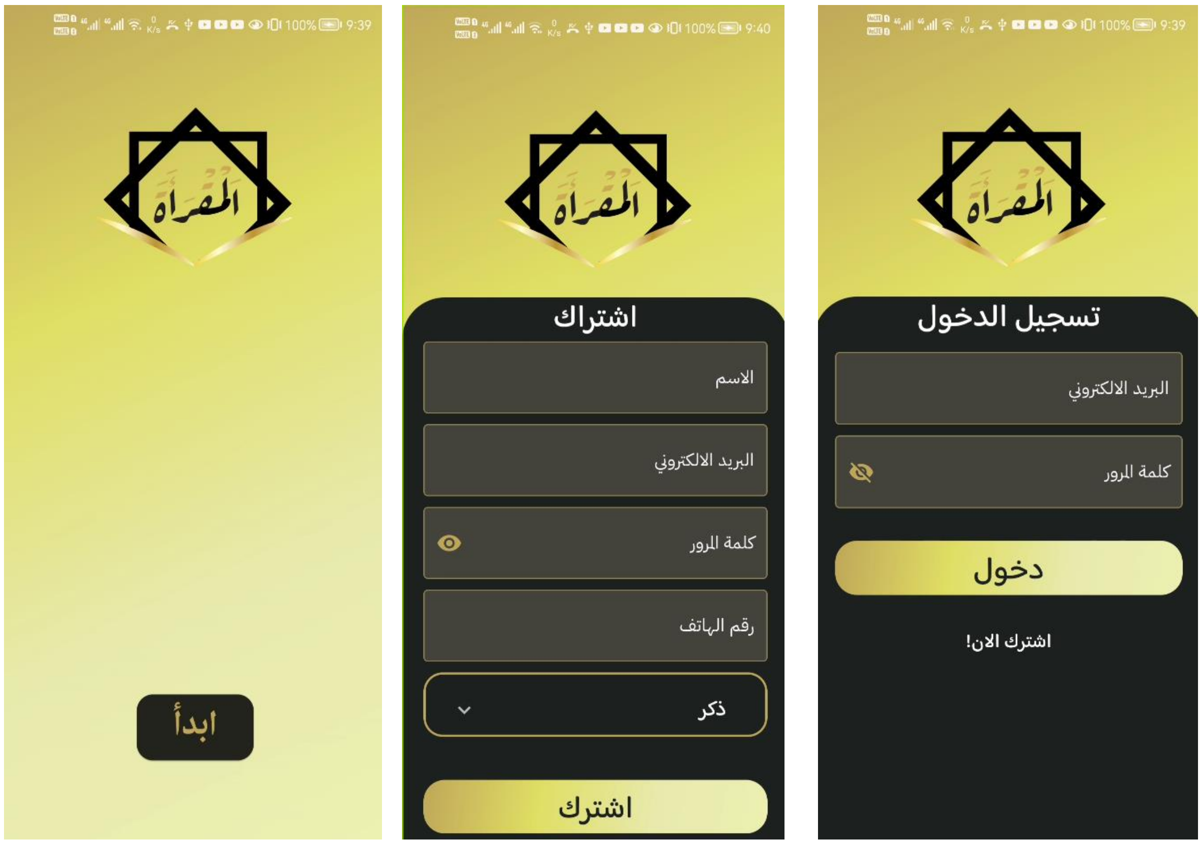


Fig . 13 Sample of System GUI - 3

Chapter 5: Approach

5.1. Data Set

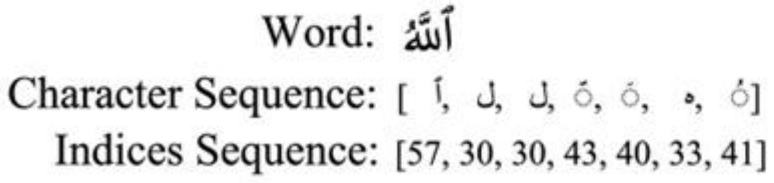
We prepared a comprehensive dataset for our study, consisting of 80,000 WAV files for the training set and 7,500 WAV files for the validation set. Each WAV file corresponds to a verse recited by a specific sheikh from the Quran, with a new sheikh beginning recitation approximately every 6,236 verses. Additionally, we generated JSON files for the training, testing, and validation sets. Each JSON file includes columns for the audio path, duration, and transcription, ensuring well-organized and accessible data for further analysis and model training.

```
{
"audio_filepath": "./testing_wav/sample_0.wav",
"duration": 23.584,
 "text": "خَتَمَ اللَّهُ عَلَى قُلُوبِهِمْ وَعَلَى سَمْعِهِمْ وَعَلَى أَبْصَارِهِمْ غِشَاوَةٌ وَلَهُمْ عَذَابٌ عَظِيمٌ"
}
{
"audio_filepath": "./testing_wav/sample_1.wav",
 "duration": 21.536,
"text": "يُخَادِعُونَ اللَّهَ وَالَّذِينَ آمَنُوا وَمَا يَخْدَعُونَ إِلَّا أَنْفُسَهُمْ وَمَا يَشْعُرُونَ"
}
```

Fig . 14 Data Set Sample

5.2. Data preparation

5.2.1 Character-Based Arabic Vectorization and Indexing



Word: ٱللَّهُ

Character Sequence: [ٱ, ل, ل, ◌ّ, ◌َ, ه, ◌ُ]

Indices Sequence: [57, 30, 30, 43, 40, 33, 41]

Fig . 15 Example on Character index

As we use character-based speech recognition models, each character in Arabic is treated as a class. The characters in our context include Arabic letters, diacritics, other symbols that affect letter pronunciation, and a space character. Each character is vectorized with a specific index. Thus, character sequences in any verse (i.e., transcript) are converted to sequences of indices, as exemplified in the provided table.

Character	Index	Character	Index
Empty String	0	م	31
ء	1	ن	32
أ	2	ه	33
ؤ	3	و	34
إ	4	ى	35
ئ	5	ي	36
ا	6	◌ً (Tanween Fat'h)	37
ب	7	◌ٌ (Tanween Dham)	38
ة	8	◌ٍ (Tanween Kaser)	39
ت	9	◌َ (Fat'ha)	40
ث	10	◌ُ (Dhamma)	41
ج	11	◌ِ (Kasra)	42
ح	12	◌ّ (Shadda)	43
خ	13	◌ْ (Sekoon)	44
د	14	◌ٓ (Extending Madd)	45
ذ	15	◌ٔ (Hamza Musaghara)	46
ر	16	◌ۜ (Pronouncing it as Seen is preferred than Sadd)	47
ز	17	◌۟ (Alsufr Almustatel)	48
س	18	◌۠ (Alsufr Almustadeer)	49
ش	19	◌ۢ (Iqlab)	50
ص	20	◌ۣ (Pronouncing it as Sadd is preferred than Seen)	51
ض	21	ۥ (Small Waw)	52
ط	22	ۦ (Small Yaa)	53
ظ	23	◌ۨ (Small Noon)	54
ع	24	◌ٰ (Almaddia Alsaghera)	55
غ	25	◌۪ (Emalah)	56
ـ	26	ٱ (Hamzat Wasel)	57
ف	27	◌۫ (Ebdal)	58
ق	28	◌۬ (Tas'heel)	59
ك	29	◌ۭ (Iqlab)	60
ل	30	Space	61

Table 3 - Character-Based Arabic Vectorization and Indexing

5.2.2. Automated Conversion and Preprocessing of Audio Files

This Python script is designed to convert a series of audio files from a specified input directory to PCM format, and subsequently preparing these converted files for use in a training dataset. The script includes a `convert_to_pcm` function to read each audio file, set its frame rate to 16kHz, convert it to mono (PCM format 1), and export it in WAV format. The `convert_files_to_pcm` function iterates over a range of files, applying the conversion, and reports on any missing or unconvertible files. The script then defines a `decode_audio` function to read and decode WAV files, and a `get_audio_dataset` function to create a dataset from the audio files in the output directory. Finally, the script processes a batch of audio data from this dataset, printing a spectrogram of the processed audio data.

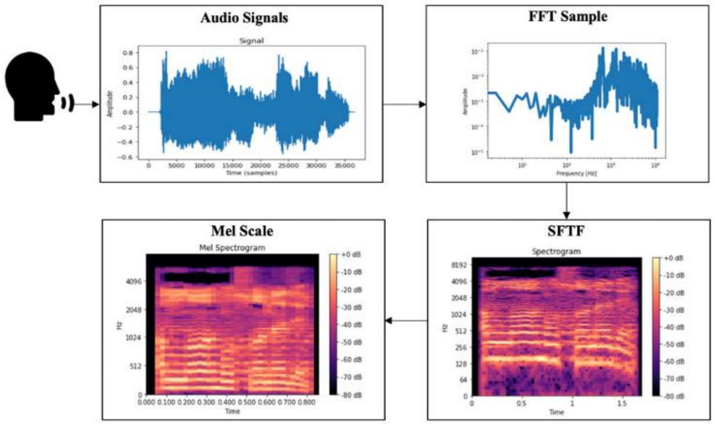


Fig . 16 from audio to mel-spectogram

```python
from pydub import AudioSegment
import tensorflow as tf
import os

def convert_to_pcm(input_path, output_path):
    try:
        audio = AudioSegment.from_file(input_path)
        audio = audio.set_frame_rate(16000)
        audio = audio.set_channels(1)
        audio.export(output_path, format="wav")
        print(f"Successfully converted {input_path} to {output_path}")
    except Exception as e:
        print(f"Failed to convert {input_path}. Error: {e}")

def convert_files_to_pcm(start, end, input_dir, output_dir):
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    all_files_converted = True
    for i in range(start, end + 1):
        filename = f"sample_{i}.wav"
        input_path = os.path.join(input_dir, filename)
        output_path = os.path.join(output_dir, filename)
        if os.path.exists(input_path):
            convert_to_pcm(input_path, output_path)
        else:
            print(f"File {input_path} does not exist.")
            all_files_converted = False
```

```
input_directory = "I:\\Sallam\\Almaqraa\\training_wav"
output_directory = "D:\\data"

convert_files_to_pcm(0, 57500, input_directory, output_directory)

def decode_audio(filename):
    audio_binary = tf.io.read_file(filename)
    audio, _ = tf.audio.decode_wav(audio_binary)
    return audio

def get_audio_dataset(directory):
    files = tf.io.gfile.glob(os.path.join(directory, "*.wav"))
    dataset = tf.data.Dataset.from_tensor_slices(files)
    dataset = dataset.map(decode_audio)
    return dataset
```

Table 4 - from audio to mel-spectogram

5.2.3. Signal Pre-emphasis

`Preemphasis` is a technique used to enhance the signal-to-noise ratio, particularly at higher frequencies. It involves amplifying the high-frequency components of the audio signal. The `preemphasis` function implements this by applying a filter that subtracts a fraction of the previous signal value from the current signal value.

```
def preemphasis(signal, preemphasis_coeff=0.97):
    return np.append(signal[0], signal[1:] - preemphasis_coeff * signal[:-1])
```

Table 5 - Signal Pre-emphasis

5.2.4. Silence Removal

Removing silence from audio recordings can significantly reduce the amount of data that needs to be processed, improving both the efficiency and accuracy of speech recognition models. This is done by detecting and trimming silence at the beginning and end of the audio files, and merging short pauses with non-silent segments.

The `merge_segments` function merges non-silent segments with short pauses between them, ensuring that brief silences do not fragment the audio unnecessarily.

The `remove_silence` function trim silent parts of the audio based on an amplitude threshold and merges short pauses with non-silent segments.

```
def merge_segments(intervals, min_silence_duration=0.5, sample_rate=None):
    merged_intervals = []
    prev_end = None
    for start, end in intervals:
        if prev_end is None:
            prev_end = end
            merged_intervals.append((start, end))
        else:
            silence_duration = (start - prev_end) / sample_rate
            if silence_duration < min_silence_duration:
                merged_intervals[-1] = (merged_intervals[-1][0], end)
            else:
                merged_intervals.append((start, end))
            prev_end = end
    return merged_intervals

def remove_silence(audio, sample_rate, top_db=30, min_silence_duration=0.5):
    non_silent_intervals = librosa.effects.split(audio, top_db=top_db)

    merged_intervals = merge_segments(non_silent_intervals,
min_silence_duration, sample_rate)

    segments = [audio[start:end] for start, end in merged_intervals]

    processed_audio = np.concatenate(segments)

    return processed_audio.astype(np.float32)
```

Table 6 - Silence Removal

5.2.5. Echo Noise Reduction

Echo noise in audio recordings can be distracting and detrimental to the performance of speech recognition models. The `spectral_subtraction` technique helps in reducing echo noise by subtracting a noise spectrum from the audio signal's magnitude spectrum.

```
def spectral_subtraction(audio_data, alpha=2.0):
    stft_matrix = librosa.stft(audio_data)
    magnitude = np.abs(stft_matrix)
    phase = np.angle(stft_matrix)
    noise_spectrum = np.median(magnitude, axis=1)
    clean_magnitude = np.maximum(magnitude - alpha * noise_spectrum[:,
np.newaxis], 0)
    clean_stft = clean_magnitude * np.exp(1j * phase)
    clean_audio = librosa.istft(clean_stft)
    return clean_audio
```

Table 7 - Echo Noise Reduction

5.2.6. Comprehensive Preprocessing Pipeline

The `preprocessing` function combines all the aforementioned steps into a single pipeline. It applies spectral subtraction, preemphasis, and silence removal sequentially to clean the audio data.

```
def preprocessing(audio_data, sr, row_number):
    cleaned_audio = spectral_subtraction(audio_data)
    cleaned_audio = preemphasis(cleaned_audio)
    cleaned_audio = remove_silence(cleaned_audio, sr)
    output_file = os.path.join('training_wav/', f"sample_{row_number}.wav")
    write(output_file, sr, cleaned_audio)
    return cleaned_audio
```

Table 8 - Comprehensive Preprocessing Pipeline

5.3. Model Architecture

5.3.1. TDNN Speech Recognition Model with CTC

Overview

Time Delay Neural Networks (TDNN) have been extensively used as acoustic models in traditional and hybrid speech recognition systems. With the advent of end-to-end speech recognition models, combinations of Convolutional Neural Networks (CNN) and TDNN have demonstrated state-of-the-art results. One notable model is Jasper (Just Another SPEech Recognizer), a TDNN-CTC end-to-end speech recognition model developed by NVIDIA in 2019. Jasper employs Connectionist Temporal Classification (CTC) loss and features a block architecture, with each block consisting of convolutional sub-blocks containing four layers. These blocks are interconnected using residual connections, which allow data to flow through different paths, potentially skipping some layers to reach the final layer. This differs from sequential connections where data flows in a single path constructing feedforward neural networks.

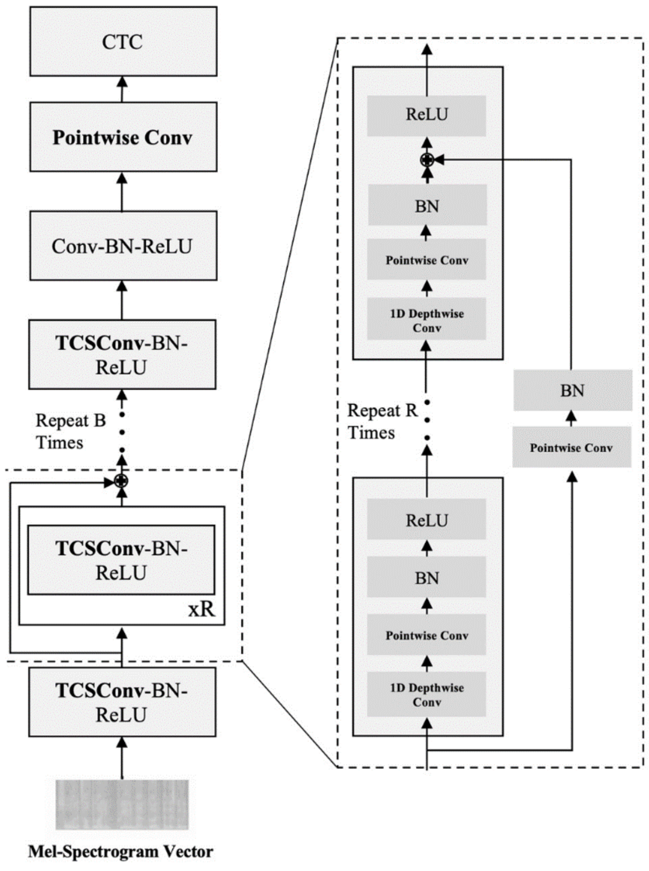


Fig . 17 QuartzNet

Jasper and QuartzNet

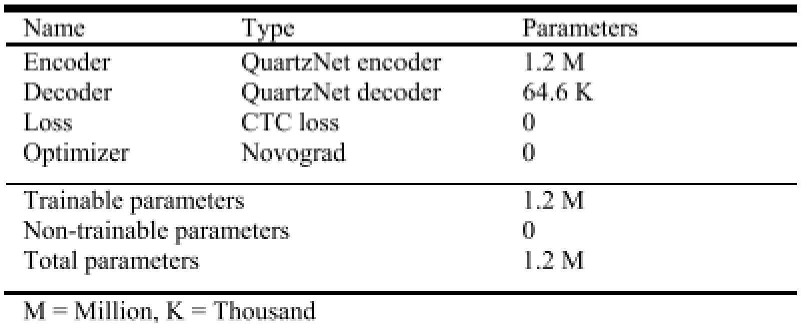
Jasper achieved state-of-the-art results on English speech datasets but requires high computational power and memory due to its extensive use of parameters, exceeding 200 million. To address this, a smaller model called QuartzNet was proposed, based on the Jasper architecture but with fewer parameters and lower computational requirements. QuartzNet utilizes depthwise separable convolutions, replacing Jasper’s 1D convolutions with 1D time-channel separable convolutions. This method handles spatial (height and width) and depth (channels) dimensions, speeding up the network and reducing complexity by splitting the kernel into depthwise and pointwise convolutions. Depthwise convolution operates on each channel across time frames, while pointwise convolution operates independently on each time frame across all channels.

Application to Classic Arabic Speech

In this work, QuartzNet is applied as a TDNN-CTC speech recognition model to recognize classic Arabic speech, marking its first use in this context. QuartzNet's components and architecture are illustrated in Figure 5.

Experimental Setup

Our TDNN model consists of an encoder, decoder, and CTC loss. We utilized the default QuartzNet encoder architecture provided by NeMo, which includes two fixed blocks (first and last blocks) and 15 repeated blocks, each containing five sub-blocks. The encoder in the QuartzNet model is based on the Jasper model and comprises seven Jasper layers (six residual layers and one traditional layer). In contrast, the decoder is a linear classification layer that converts the encoder's output into probabilities for 63 classes (62 characters plus one blank character for the CTC loss). The CTC loss is then calculated to map the highest probability class to its character representation using a greedy decoder.



Name	Type	Parameters
Encoder	QuartzNet encoder	1.2 M
Decoder	QuartzNet decoder	64.6 K
Loss	CTC loss	0
Optimizer	Novograd	0
Trainable parameters		1.2 M
Non-trainable parameters		0
Total parameters		1.2 M

M = Million, K = Thousand

Table 9 QuartzNet Parameters

Optimization

Our TDNN model employs the Novograd optimization method, an adaptive layer-wise stochastic optimization technique that normalizes gradients and decouples weight decay per layer.

5.3.2. RNN Speech Recognition Model with CTC

Overview

Recurrent Neural Networks (RNN) have been effectively utilized in end-to-end speech recognition models to convert audio spectrograms into text transcriptions. A notable example is the Deep Speech model, which comprises five layers of hidden units: the first three and the last layers are non-recurrent, while the fourth layer is an RNN that processes the data in both forward and backward directions. This model uses Connectionist Temporal Classification (CTC) loss to align the encoder's output with character sequences. However, models with a single recurrent layer in the encoder struggle to handle large and continuous speech datasets, limiting their performance.

Deep Speech 2 Enhancements

To overcome these limitations, the Deep Speech 2 model was introduced, incorporating multiple CNN and RNN layers. This enhanced version, which includes one CNN layer, five Gated Recurrent Unit (GRU) layers, and one fully connected layer, achieved the lowest Word Error Rate (WER) compared to other combinations of CNN and RNN layers. Inspired by these improvements, we constructed our RNN-CTC speech recognition model based on the enhanced Deep Speech 2 architecture, which has shown superior recognition performance.

Model Architecture

Our RNN-CTC speech recognition model consists of the following components:

Encoder: The encoder comprises four CNN layers (one traditional CNN and three Residual CNN layers), five Bidirectional GRU (BiGRU) layers, and one fully connected layer. The CNN layers extract audio features, while the BiGRU layers predict each frame's output, considering the information from previous fr ames. GRU layers, a variant of RNN, require fewer computational resources compared to Long Short-Term Memory (LSTM) layers.

Decoder: The decoder is a linear classification layer that converts the encoder's output into probabilities for 63 classes (62 characters and one blank character for the CTC loss).

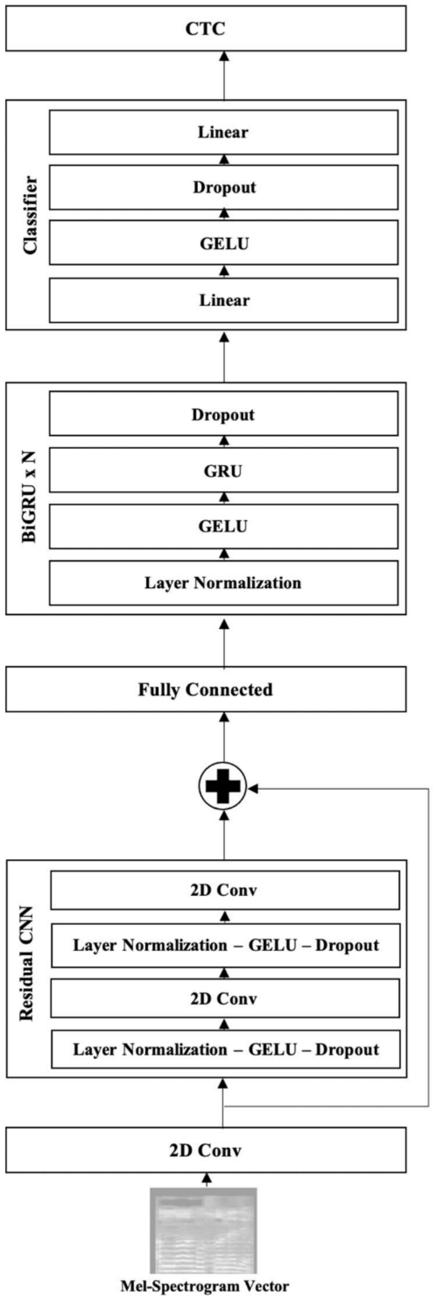


Fig . 18 RNN-CTC

CTC Loss: The CTC loss function aligns the predicted character probabilities with the actual character sequences using a greedy decoder.

The architecture of our RNN-CTC model is illustrated in Figure 6.

Optimization

Our RNN model employs the AdamW optimization method, a modified version of the Adam optimizer that decouples weight decay from the gradient updates. This method is effective in improving the convergence and performance of the model.

Name	Type	Parameters
Encoder	RNN encoder	26.6 M
Decoder	Sequential decoder	0
Loss	CTC loss	0
Optimizer	AdamW	0
Trainable parameters		26.6 M
Non-trainable parameters		0
Total parameters		26.6 M

M = Million, K = Thousand

Table 10 - RNN CTC Parameters

2.3.3. Experimental Details

We conducted experiments to evaluate the performance of our TDNN-CTC and RNN-CTC speech recognition models. The models were trained with different configurations and optimized using appropriate optimization methods to ensure optimal performance. The details of the experimental setup for each model are as follows:

Models	Epochs	Batch size	Learning rate	Optimizer	Accuracy
TDNN-CTC	168	32	0.01	Novgard	50%
RNN-CTC	20	8	0.005	AdamW	70%

Table 11 Experimental Details

The experimental results highlight the superiority of the RNN-CTC model over the TDNN-CTC model in terms of accuracy. The advanced architecture of the RNN-CTC model, which includes CNN and BiGRU layers, and the use of the AdamW optimizer contributed to its enhanced performance. These findings underscore the importance of model architecture and optimization techniques in developing effective speech recognition systems

Chapter 6: Implementation and testing

6.1 Challenges

First Challenge: Computational Power

One of the significant challenges faced during the development of the Al-Maqraa application is the issue of computational power. The application relies heavily on advanced machine learning models, specifically CNN-CTC and RNN-CTC, to evaluate the accuracy of Quranic recitations. These models require substantial computational resources for training and real-time inference, which poses several challenges:

- Training deep learning models like CNN-CTC and RNN-CTC is computationally intensive and demands high-performance hardware. However, access to such hardware is often limited, leading to prolonged training times and difficulties in experimenting with model architectures and hyperparameters.

- For the application to provide immediate feedback on recitations, it must perform real-time inference. This requirement necessitates a robust computational infrastructure capable of handling numerous simultaneous requests without compromising on speed or accuracy. Limited computational power can result in delays, reducing the effectiveness and user experience of the application.

- As the user base grows, the computational demands will increase correspondingly. Ensuring that the system can scale efficiently to handle more users while maintaining performance levels is a significant challenge. Limited computational resources can hinder the application's ability to scale, affecting its accessibility and reliability.

- High-performance computing resources are not only limited but also expensive. Balancing the need for computational power with budget constraints is a continual challenge. Investing in the necessary infrastructure to support the application's requirements can be costly, especially during the initial development phases.

Second Challenge:

Another significant challenge is dealing with cloud computing and High-Performance Computing (HPC) resources provided by Bibliotheca Alexandrina. While leveraging cloud and HPC resources is essential for scaling and handling intensive computations, several issues have arisen:

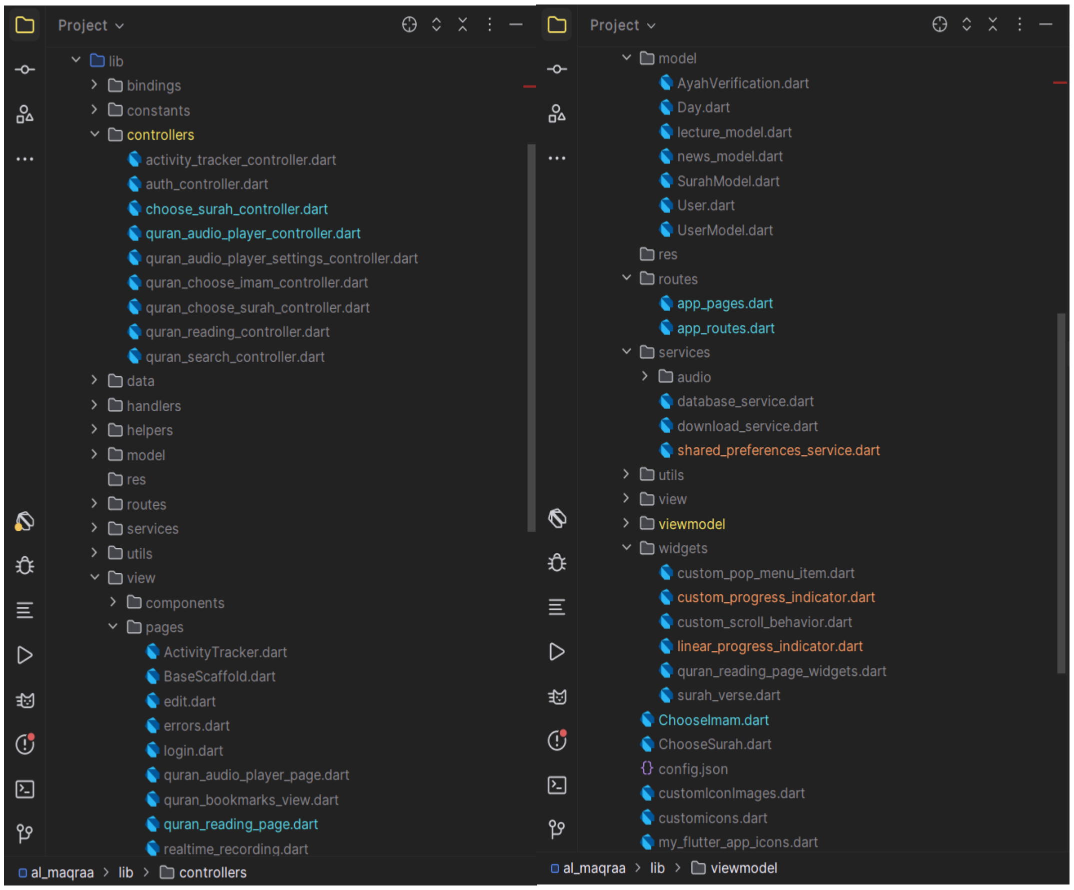
- Effective communication with the team at Bibliotheca Alexandrina has been challenging. Delays in responses and unclear communication channels have hindered the timely resolution of issues and coordination efforts. This lack of effective communication complicates the collaboration process and delays project timelines.

- The information provided by Bibliotheca Alexandrina regarding their HPC resources has often been insufficient and sometimes incorrect. This has led to difficulties in accurately assessing the available resources, planning computational tasks, and optimizing the use of their infrastructure. Misinformation further exacerbates the challenge of managing computational workloads effectively.

- The HPC machines at Bibliotheca Alexandrina are quite old and may not meet the performance requirements needed for training and deploying advanced ML models. The outdated hardware results in slower processing times and may not support the latest technologies and frameworks essential for efficient machine learning tasks.

6.2. Implementation

6.2.1. Frontend



```
Project
lib
  bindings
  constants
  controllers
    activity_tracker_controller.dart
    auth_controller.dart
    choose_surah_controller.dart
    quran_audio_player_controller.dart
    quran_audio_player_settings_controller.dart
    quran_choose_imam_controller.dart
    quran_choose_surah_controller.dart
    quran_reading_controller.dart
    quran_search_controller.dart
  data
  handlers
  helpers
  model
  res
  routes
  services
  utils
  view
    components
    pages
      ActivityTracker.dart
      BaseScaffold.dart
      edit.dart
      errors.dart
      login.dart
      quran_audio_player_page.dart
      quran_bookmarks_view.dart
      quran_reading_page.dart
      realtime_recording.dart
al_maqraa > lib > controllers
```

```
Project
  model
    AyahVerification.dart
    Day.dart
    lecture_model.dart
    news_model.dart
    SurahModel.dart
    User.dart
    UserModel.dart
  res
  routes
    app_pages.dart
    app_routes.dart
  services
    audio
    database_service.dart
    download_service.dart
    shared_preferences_service.dart
  utils
  view
  viewmodel
  widgets
    custom_pop_menu_item.dart
    custom_progress_indicator.dart
    custom_scroll_behavior.dart
    linear_progress_indicator.dart
    quran_reading_page_widgets.dart
    surah_verse.dart
  ChooseImam.dart
  ChooseSurah.dart
  config.json
  customIconImages.dart
  customicons.dart
  my_flutter_app_icons.dart
al_maqraa > lib > viewmodel
```

Fig . 19 - Frontend Package

To ensure our codebase is clean, readable, and adheres to the principle of single responsibility, we organized our Dart files into several key directories: `model`, `routes`, `services`, `view`, `viewmodel`, `controllers`, and `pages`.

Model: This directory contains data structures that represent the application's core concepts, facilitating data transfer across different parts of the application. Examples include `UserModel.dart`, `SurahModel.dart`, and `news_model.dart`.

Routes: This folder holds files related to the navigation of the application, such as `app_pages.dart` and `app_routes.dart`, defining the app's route management.

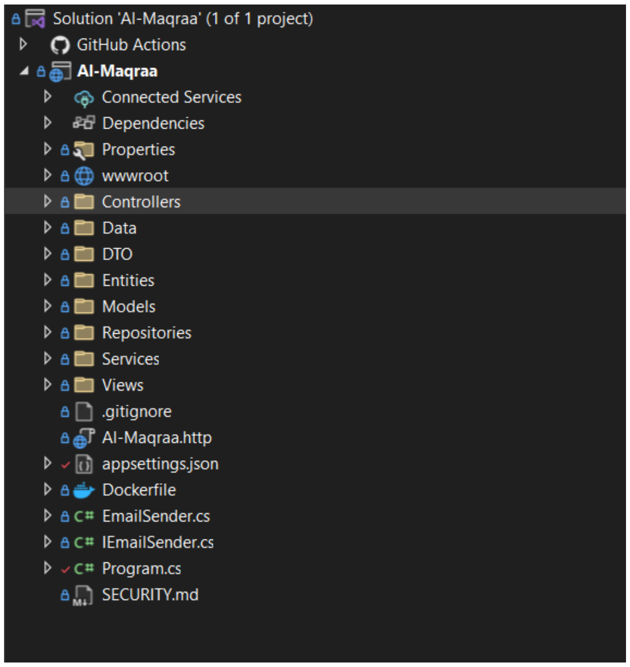
Services: This directory includes service files that handle specific functionalities like database interactions (`database_service.dart`), audio processing (`audio` folder), and shared preferences management (`shared_preferences_service.dart`).

View: It consists of the user interface components for the application, containing various widgets used across different pages. The `viewmodel` subfolder includes files that bridge the gap between the view and model, ensuring that the UI components correctly display the data.

Controllers: These files manage the business logic, handling user inputs, validation, and communication with the backend services. Examples are `auth_controller.dart`, `quran_audio_player_controller.dart`, and `activity_tracker_controller.dart`.
Pages: This folder contains the actual design and layout files for both mobile and web applications. Each file represents a distinct page or screen in the app, such as `login.dart`, `quran_reading_page.dart`, and `ActivityTracker.dart`.
By maintaining this structured approach, we enhance code reusability and maintainability, making it easier for developers to navigate and manage the codebase efficiently.

6.3.1. Backend

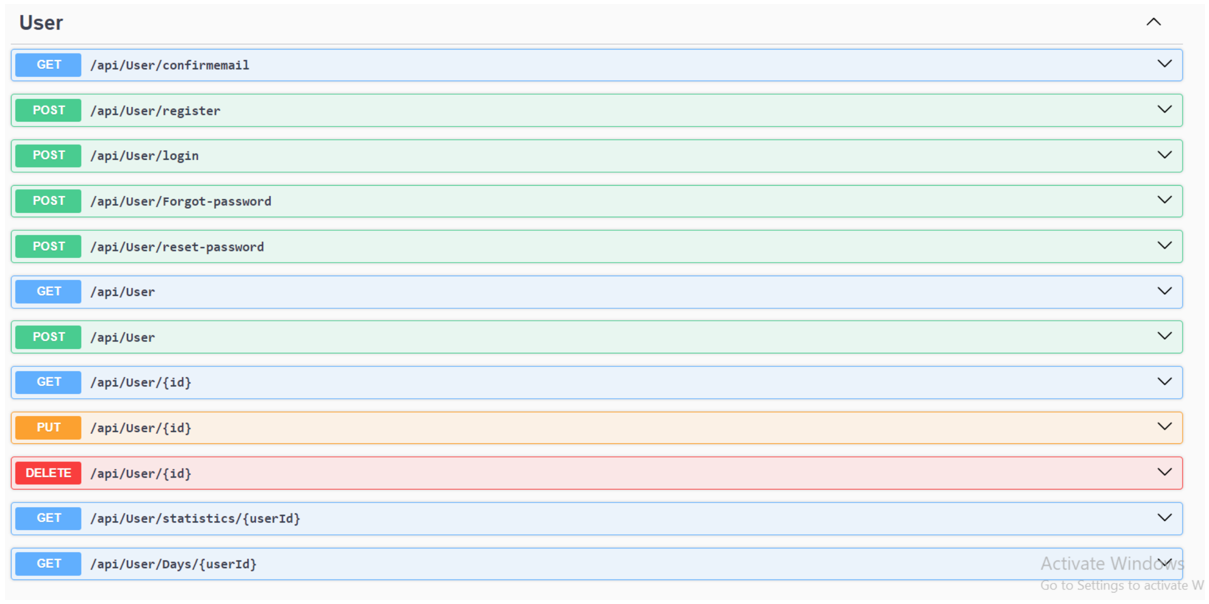
Back-end As stated before, we used ASP.NET framework to develop our APIs and the backend. just as in the class diagram, we divided our system into packages, each package contains the entities/models needed, serializers and for each major model, there is a controller that handles the functions related to it. The controllers communicate with each other to perform the overall service. The serializers transform the object into a json format so that we could send this json to the frontend to use the data.



```
Solution 'Al-Maqraa' (1 of 1 project)
  GitHub Actions
  Al-Maqraa
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
    Data
    DTO
    Entities
    Models
    Repositories
    Services
    Views
    .gitignore
    Al-Maqraa.http
    appsettings.json
    Dockerfile
    EmailSender.cs
    IEmailSender.cs
    Program.cs
    SECURITY.md
```

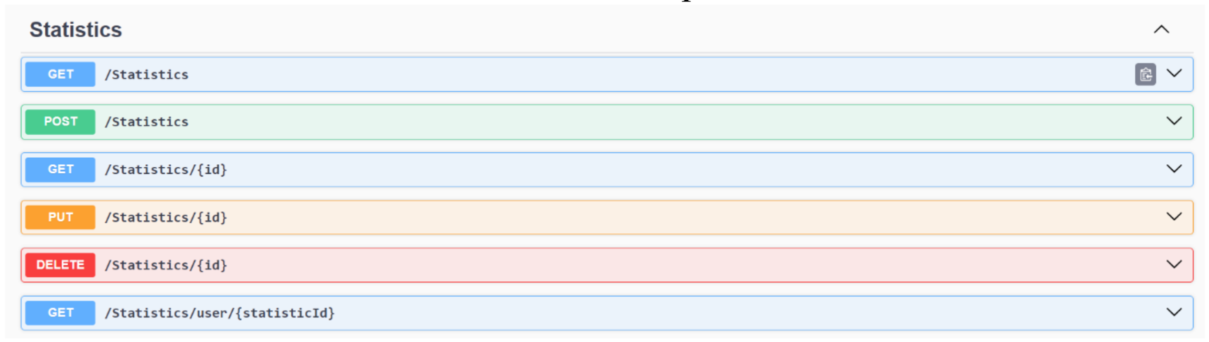
Fig . 20 Backend Package

This is our end points, and we use swagger to test it


User

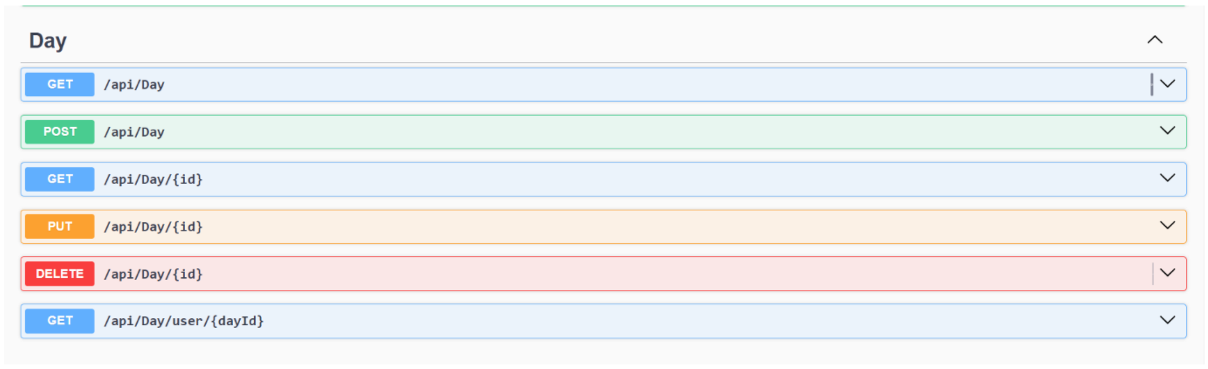
GET /api/User/confirmemail
POST /api/User/register
POST /api/User/login
POST /api/User/Forgot-password
POST /api/User/reset-password
GET /api/User
POST /api/User
GET /api/User/{id}
PUT /api/User/{id}
DELETE /api/User/{id}
GET /api/User/statistics/{userId}
GET /api/User/Days/{userId}

User model end points


Statistics

GET /Statistics
POST /Statistics
GET /Statistics/{id}
PUT /Statistics/{id}
DELETE /Statistics/{id}
GET /Statistics/user/{statisticId}

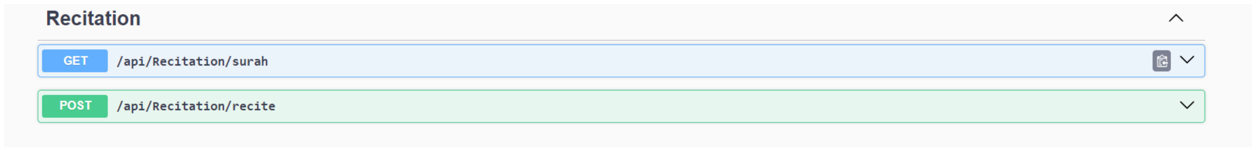
Statistics of User model endpoints


Day

GET /api/Day
POST /api/Day
GET /api/Day/{id}
PUT /api/Day/{id}
DELETE /api/Day/{id}
GET /api/Day/user/{dayId}

Day model to mark the daily usage of the User endpoints

Recitation Service end points that take the ayah from the front end and take the recited ayah from the ML model and then specify the error between then and return it to the frontend


Recitation

GET /api/Recitation/surah
POST /api/Recitation/recite

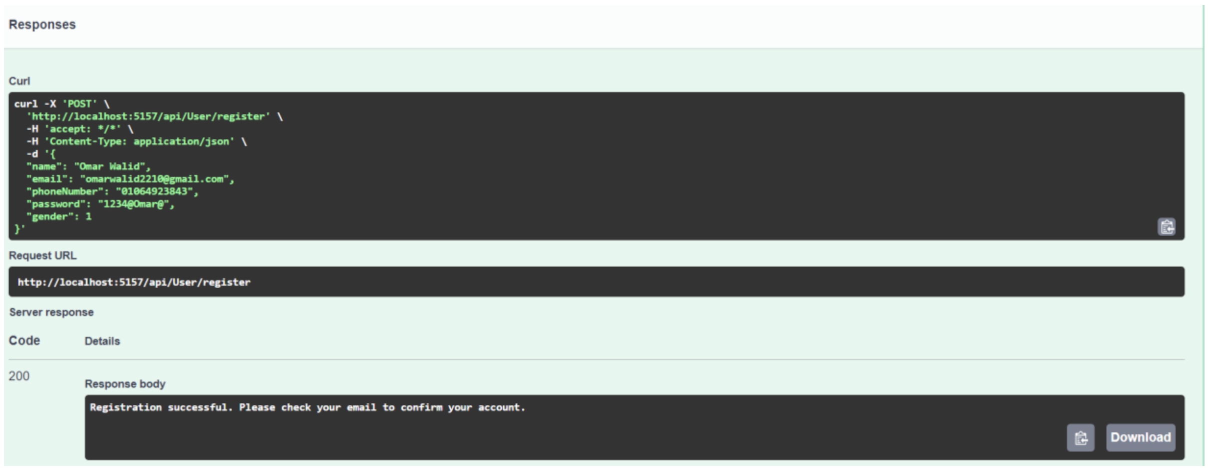
6.3. Testing:

We tested our system from Swagger and Postman to test the APIs, the frontend to make sure that the code is run correctly.

6.3.1. Testing endpoints of the User Model

1. Test Register User with correct data:

The User registration Success but he should confirm his email



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5157/api/User/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Omar Walid",
  "email": "omarwalid2210@gmail.com",
  "phoneNumber": "01064923843",
  "password": "1234@Omar@",
  "gender": 1
}'
```

Request URL

```
http://localhost:5157/api/User/register
```

Server response

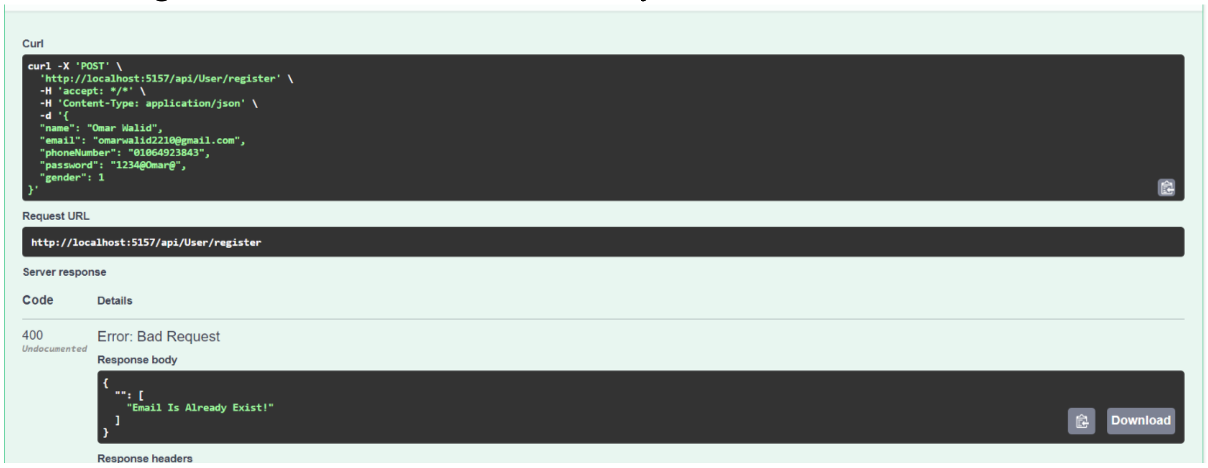
Code Details

200

Response body

```
Registration successful. Please check your email to confirm your account.
```

2. Test Register User with the email already exist



Curl

```
curl -X 'POST' \
  'http://localhost:5157/api/User/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Omar Walid",
  "email": "omarwalid2210@gmail.com",
  "phoneNumber": "01064923843",
  "password": "1234@Omar@",
  "gender": 1
}'
```

Request URL

```
http://localhost:5157/api/User/register
```

Server response

Code Details

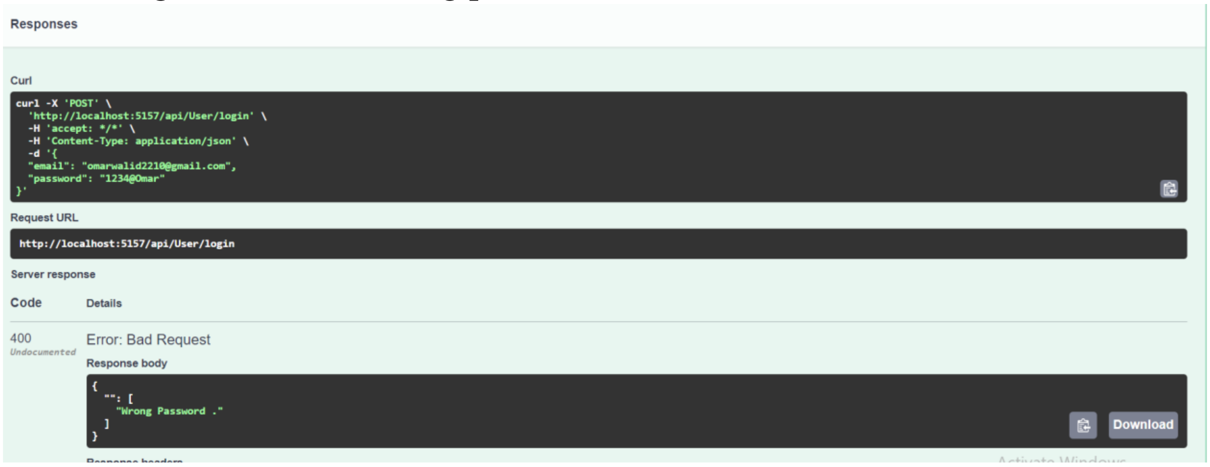
400 *Undocumented* Error: Bad Request

Response body

```
{
  "": [
    "Email Is Already Exist!"
  ]
}
```

Response headers

3. Test Login User with wrong password

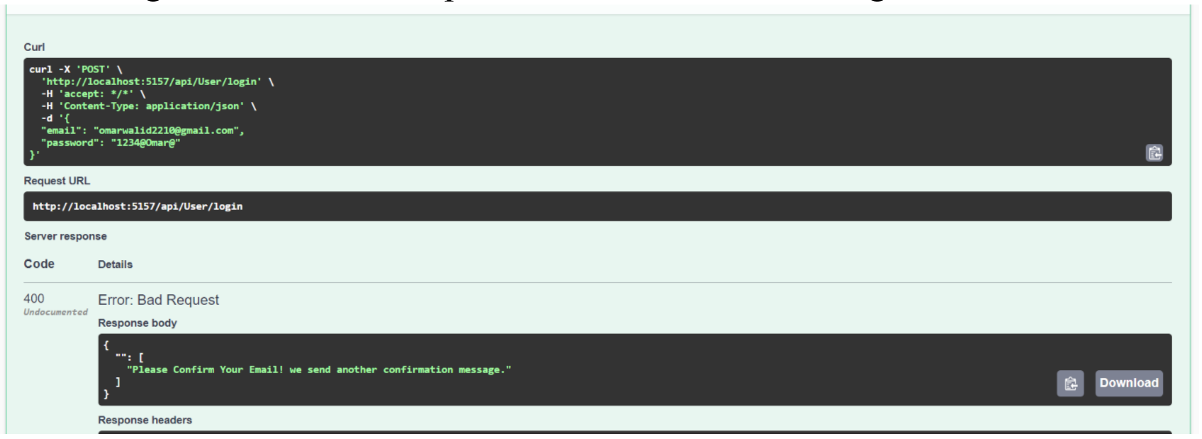


```
curl -X 'POST' \
  'http://localhost:5157/api/User/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "omarwalid2210@gmail.com",
  "password": "1234@Omar"
}'
```

```
http://localhost:5157/api/User/login
```

```
{
  "": [
    "Wrong Password ."
  ]
}
```

4. Test Login with the correct password but without making confirm for the email

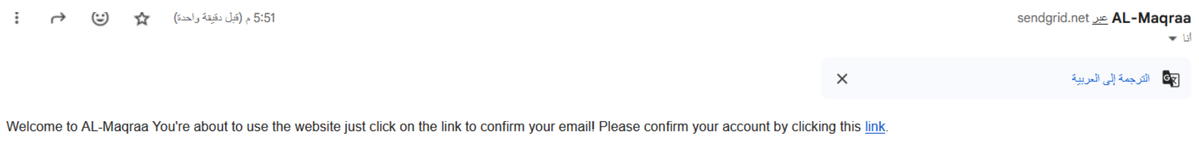


```
curl -X 'POST' \
  'http://localhost:5157/api/User/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "omarwalid2210@gmail.com",
  "password": "1234@Omar@"
}'
```

```
http://localhost:5157/api/User/login
```

```
{
  "": [
    "Please Confirm Your Email! we send another confirmation message."
  ]
}
```

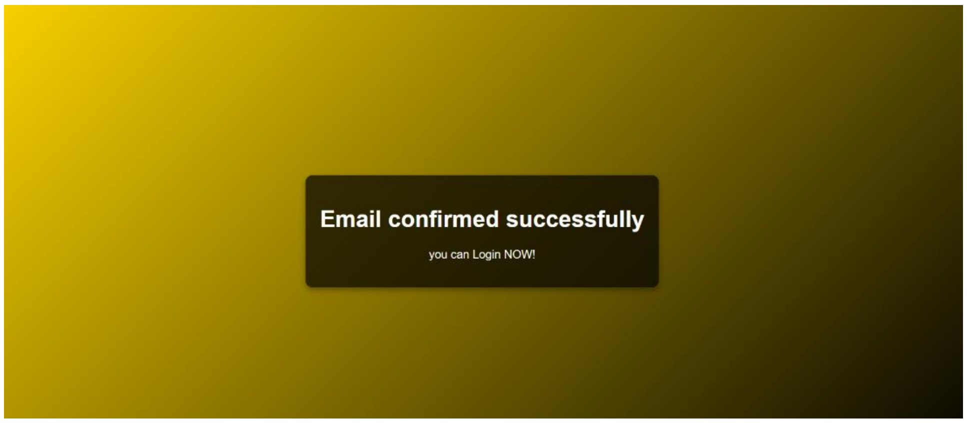
- After click on the link in the email received it will be confirmed



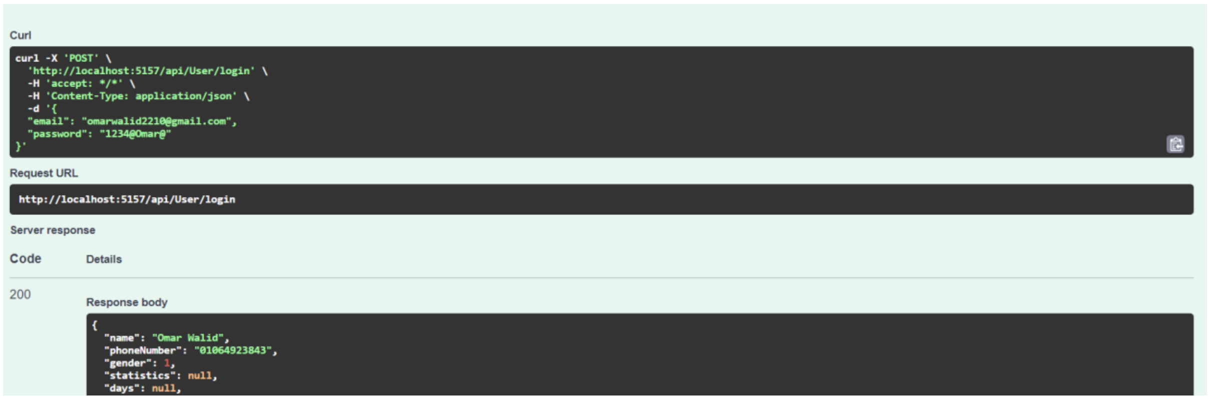
Welcome to AL-Maqraa You're about to use the website just click on the link to confirm your email! Please confirm your account by clicking this link.

-

This will be received after confirmation and the user can login



5. Test Login with the correct password and with making confirm for the email



Curl

```
curl -X 'POST' \
  'http://localhost:5157/api/User/login' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "omarwalid2210@gmail.com",
  "password": "1234@Omar@"
}'
```

Request URL

```
http://localhost:5157/api/User/login
```

Server response

Code	Details
200	Response body

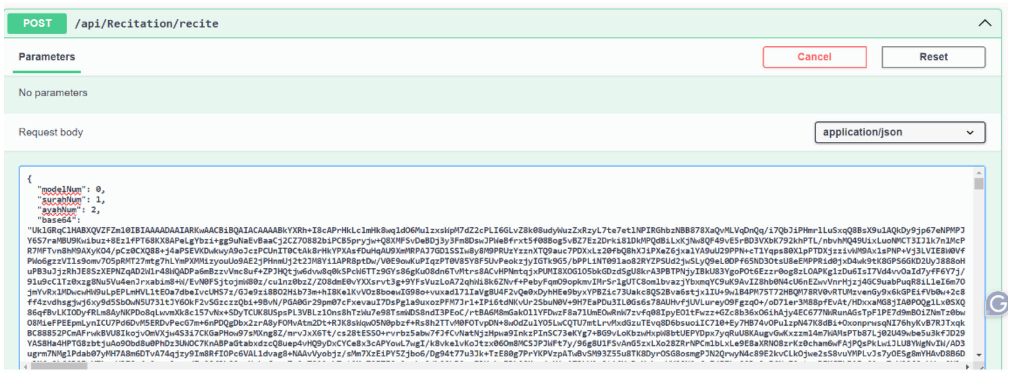
```
{
  "name": "Omar Walid",
  "phoneNumber": "01064923843",
  "gender": 1,
  "statistics": null,
  "days": null,
```

6.3.2. Testing endpoints of the Recitation service

We should send in POST request to the recitation endpoint have JSON that should take the model number, surah number, ayah number and base64 is the encoded format of the audio of the recitation ayah of the user and we will get a response with the mistakes and if there is no mistakes the all words will be true in the response .

Example by using audio of the “الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ” ayah

The request body



POST /api/Recitation/recite

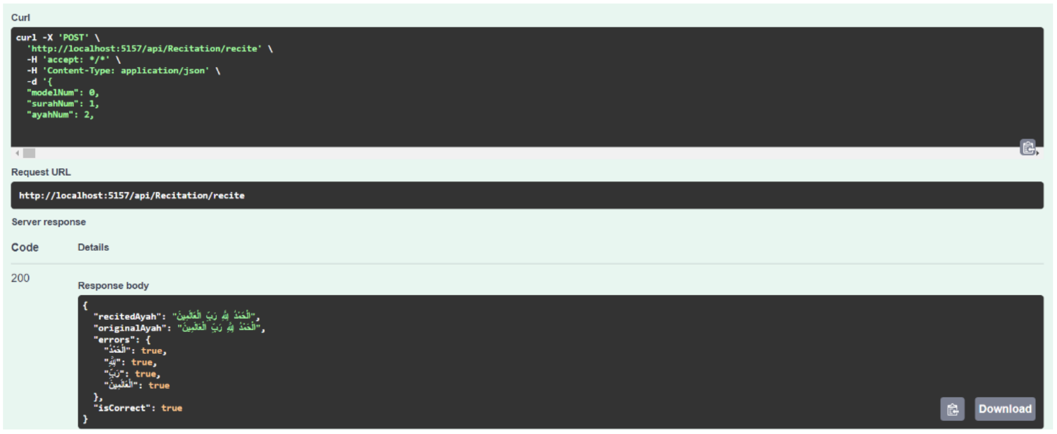
Parameters

No parameters

Request body: application/json

```
{
  "modelNum": 0,
  "surahNum": 1,
  "ayahNum": 2,
  "base64":
"UklGRqClHABXQVZFZm10IBIAAAADAAIARKwAACBiBQAIACAAAABkYXRh+I8cAPrHkLclmHk8wq1dO6MulzxsWpM7dZ2cPLI6GLvZ8k08udyWuzZxRzyL7te7etlNPIRGhbzNBB878XaQvMLVqDnQq/i7QbJiPHmrlLuSxqQ8BsX9u1AQkDy9jp67eNPMPJ
Y6S7raMBU9Kwibuz+8Ez1fPT68KX8APeLgYbzi+gg9uNaEvBaaCj2CZ7O882biPCB5pryjw+Q8XMFSvDeBDj3y3Fm8DswJPWeBfrxt5f08Bog5vBZ7Ez2Drki81DkMPQdBiLxKjNw8QF49vE5rBD3VXbK792khPTL/nbvhMQ49UixLuoNMCT3IJ1k7n1McP
R7MFTvn8hM9AXyKO4/pCz0CXQ88+j4aPSEVKDwkwyA9oJczPCUnIT0CtAk8rHkYPXAsfDuHqAU9XmMRPAJ7GD1SSIw8y8M9PRUzYzznXTQ9auc7PDXxLz20fbQ8hXJiPXeZ6jxalYA9uU29PPN+cT1Yqps80XlpPTDXjzzivWM9Ax1sPNP+Vj3LVIE8W0Vf
PWo6gzzVIls9omv7O5pRMT27mtg7hLYmPXMMizyouUo9AE2jPHnmUj2t2JM8Yi1APR8ptDw/V0E9owKuPIqzPT0V85Y8F5UvPeokzjyIGTk9G5/bPPLiNT091ao82RYZPSUd2jwSLyQ9eL0DPf65ND3OtsU8eEMPPRid0jxD4wk9tK8GPS6GKD2UyJ888oH
uPB3uJjzRhJE8SzXEPNZqAD2W1r48WQADPa6mBzzvVmc8uf+ZPJHQtjw6dvw8q0kSPcW6TTz9GYs86gKuO8dn6TvMtrs8ACvHPNmtqjxPUMI8XOG1O5bkGDzdSgU8krA3PBTPNjyIBkU83YgoPOt6Ezzr0og8zLOAPKg1zDu6IsI7Vd4vvOaId7yfF6Y7j/
91u9cClTz0xzg8Nu5Vu4enJrxabim8+W/EvN0FSjtojmW80z/culnz0bzZ/ZO8dmE0vYXXsrvt3g+9YFsVuzLoA72qhWi8k6ZNvf+PebyFqmO9opkmvIMrSr1gUTC8omlbvazjYbxmqYC9uK9AvIZ8hb0N4cU6nEZwvVnrHjzj4GC9uabPuqR8iLleI6m7O
jmYvRx1MDwcwHW9uLpEPLmHVL1tEOa7dbeIvcUHS7z/GJe9zi8BO2Hib73m+hI8KelKvVOz8boewIG98o+vuxad171IaVg8U4F2vQe0xDyhHEe9byxYPBZic73Uakc8QS2Bva6stjx1IU+9w1B4PM75T72HBQM78RV0vRTUMzvenGy9x6kGPEifVb0w+2c8
ff4zvdhsgjwj6xy9d5SbOwN5U731tJY6OkF2vSGzczzQbi+9BvN/PGA0Gr29pm07cFxevauI7DsPgla9uxozPFM7Jr1+IPi6tdNKvUr2SbuN0V+9H7EaPDu3IL0Gs6s78AUHvfjUVLureyO9FgzqO+/oD71er3M88pfEvAt/HDxxaMG8jIA0POQg1Lx0SXQ
86qfBvLKIODyfRLm8AyNKPDo8qLwvmXk8c157vNx+SDyTCUK8USpsPL3VBLz1Ons8hTzWu7e98TsmWDS8ndI3PEoC/rtBA6M8mGakO1lYFDwzF8a71UmEOwRnW7zvfq08IpyEO1tFwzz+GZc8b36xO6ihAjy4EC677NWRunAGsTpFlPE7d9mBOiZNmTz0bw
O8MieFPEEpmLynICU7Pd6DvM5ERDvPecG7m+6nPDQgDbx2zrA8yFOMvAtm2Dt+RJK8sWqwO5N0pbzf+Rs8h2TTvM0FOTvpDN+8wOdZulYO5LwCQTU7mtLrvMxdGzuTEvq8D6bsuoiIC710+Ey7HB74vOPulzpN47K8dBi+OxonprwsqNI76hyKvB7RJTxqk
BC88852PCmAFrwkBVU8IkojvOmVXjw4S3i7CKGaPHow97sMXng8Z/mrvJxX6Tt/cs28tESSO+rvrbz5abw7fJfCvNatNjzHpwa9InkzPInSC73eKYg7+BG9vLoKbzwHxpW8btUEPYDpx7yqRuU8KAugvGwKxzzm14m7WAMsPTb87Lj02U49wbe5u3kfJD29
YAS8Ha4HPTG8zbtjuAo9Obd8u0PhDz3UWOC7KnABPaGtabxdzcQ8uep4vHQ9yDxCYCe8x3cAPYowL7wgI/k8vkelvKoJtzx06Om8MCSJPJWFt7y/96g8U1FSvAnG5zxLXo28ZRrNPCm1bLxLe9E8aXRNO8zrKz0cham6wFAjPQsPkLwiJLU8YWgNvIW/AD3
ugrm7NMglPdab07yMH7A8m6DTvA74qjzy9Im8RfIOPc6VAL1dvag8+NAAvVyobjz/sMm7XzEiPY5Zjbo6/Dg94t77u3Jk+TzE80g7PrYKPVzpATwBvSM93Z55u8TK8DyrOSG8osmgPJN2QrwyN4c89E2kvCLkOjwe2sS8vuYMPLvJs7yOESg8mYHAvD8B6D
```

The response body of the above request



Curl

```
curl -X 'POST' \
  'http://localhost:5157/api/Recitation/recite' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "modelNum": 0,
  "surahNum": 1,
  "ayahNum": 2,
```

Request URL

```
http://localhost:5157/api/Recitation/recite
```

Server response

Code	Details
200	Response body

```
{
  "recitedAyah": "الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ",
  "originalAyah": "الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ",
  "errors": {
    "الْحَمْدُ": true,
    "لِلَّهِ": true,
    "رَبِّ": true,
    "الْعَالَمِينَ": true
  },
  "isCorrect": true
}
```

References

[1] *Tarteel - Recite the Quran confidently*. (n.d.). https://www.tarteel.ai/ *Accessed 10 February 2022*
[2] Tamkeen [Mobile App]. Available :
https://play.google.com/store/apps/details?id=com.mighty.tamkeen&hl=en_US Accessed 10 February 2022
[3] Tech, E.: Tasmee. (10.01) [Mobile App]. https://play.google.com/ store/apps/details?id=com.eqra.android.tasmee. Accessed *10 February 2022*
[4] Razak, Z.; Ibrahim, N.J.; Idris, M.Y.I.; Tamil, E.M.; Yakub, M.Yusoff, Z.M.Abdulrahman, N.N.: Quranic verse recitation recognition module for support in j-QAF learning: a review. Int. J. Comput. Sci. Netw. Secur. 8, 207–216 (2008)
[5] Ibrahim, N.J.; Yakub, M.; Yusoff, Z.B.M.; Razak, Z.; Salleh, R.:Improve design for automated Tajweed checking rules engine of quranic verse recitation: a review. Int. J. Quranic Res 1, 39–50 (2011)
[6] Mohammed, A., Sunar, M. S., & Hj Salam, M. S. (2015). Quranic Verses Verification using Speech Recognition Techniques. Jurnal Teknologi, 73(2). https://doi.org/10.11113/jt.v73.4200
[7] Yousfi, B.; Zeki, A.M.: Automatic speech recognition for the holy qur'an, a review. In: Proceedings of the International Conference on Data Mining, Multimedia, Image Processing
and Their Applications (ICDMMIPA), Kuala Lumpur, Malaysia, pp. 23–29 (2016)
[8] Ahmed, A.H.; Abdo, S.M.: Verification system for Quran recitation recordings. Int. J. Comput. Appl. 163(4), 6–11 (2017). https:// doi.org/ 10.5120/ ijca2017913493
[9] S. S. Alrumiah and A. A. Al-Shargabi, "A Deep Diacritics-Based Recognition Model for Arabic Speech: Quranic Verses as Case Study," in IEEE Access
[10] Alrumiah, S.S., Al-Shargabi, A.A. Intelligent Quran Recitation Recognition and Verification: Research Trends and Open Issues. Arab J Sci Eng 48, 9859–9885 (2023). https://doi.org/10.1007/s13369-022-07273-8
[11]Quran Ayat Speech to text. (2022b, September 18). Kaggle. Accessed 10 February 2024 https://www.kaggle.com/datasets/bigguyubuntu/quran-ayat-speech-to-text
[12]Osman, Hanaa & Mustafa, Sharief & Mohammad, Yusra. (2021). QDAT: A data set for Reciting the Quran.
[13] Quran.com: EveryAyah Dataset. https://everyayah.com/ (2009). Accessed 10 February 2024
[14]Yousfi, B.; Zeki, A.M.: Holy Qur'an speech recognition system Imaalah checking rule for
warsh recitation. In: 2017 IEEE 13th International Colloquium on Signal Processing & Its Applications (CSPA), pp. 258–263. IEEE, Penang, Malaysia (2017)