Date: 09/_/2024

AMBITION COLLEGE

Mid-Baneshwor, Kathmandu

SUPERVISOR'S RECOMMENDATION

I hereby recommend that the report under my supervision by Krishchal Regmi, Sujan Basnet and Sushant Regmi entitled "**Automated Interview System**" in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Information Technology be processed for evaluation.

…….

Mr. Dharmendra Thapa.

Supervisor

Department of CSIT

Ambition College

Baneshwor, Kathmandu

Tribhuvan University

Institute of Science and Technology

AMBITION COLLEGE

Baneshwor, Kathmandu, Nepal

Letter of Approval

This is to certify that this project prepared by Krishchal Regmi, Sujan Basnet and Sushant Regmi entitled "**Automated Interview System**", in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and qualify as a project for the required degree.

Evaluation Committee

.....	.....
Mr. Yubraj Dahal	**Mr. Dharmendra Thapa.**
Head of Department	**Supervisor**
Department of Computer Science and Information Technology	Department of Computer Science and Information Technology
Ambition College	**Ambition College**
Baneshwor, Kathmandu	Baneshwor, Kathmandu

.....

External

ACKNOWLEDGEMENT

We extend our deepest gratitude to all those who supported and mentored us throughout the completion of our research project on " **Automated Interview System** " under the Department of Computer Science and Information Technology.

First and foremost, we express our heartfelt appreciation to our esteemed supervisor **Dharmendra Thapa**, whose extensive knowledge and invaluable suggestions guided us at every step of the project. His unwavering support and played a significant role in shaping our project. His advice and motivation propelled us towards achieving our goals within the stipulated timeframe We are also much indebted towards his invaluable guidance and support, which played a significant role in shaping our project. His advice and motivation propelled us towards achieving our goals within the stipulated timeframe.

Our gratitude extends to our parents, **the Principal**, and **Ambition College** for their unwavering support and encouragement throughout this journey. Their belief in us provided the foundation upon which we could build and succeed.

Furthermore, we acknowledge the contributions of numerous individuals who directly or indirectly assisted us during this project for their support and assistance. Their collective efforts were instrumental in overcoming challenges and achieving our objectives and finally, we express our sincere appreciation to our friends and well-wishers for their continuous support and encouragement for their belief in our capabilities that spurred us on during moments of doubt.

With respect,

Krishchal Regmi (26274/077)

Sujan Basnet (26294/077)

Sushant Regmi (26297/077)

ABSTRACT

The Automated Interview System is a web-based platform designed to streamline the interview process through automation. This system facilitates candidate evaluation by providing relevant question recommendations based on their selected expertise, utilizing a content-based filtering algorithm. Candidates respond verbally to the recommended questions, and their responses are transcribed using the Whisper AI model for accurate speech-to-text conversion.

The system employs cosine similarity to compare transcribed answers with a predefined dataset, generating reliable and unbiased evaluation scores. With features such as user authentication, real-time processing, and secure data handling, the system ensures an intuitive and seamless experience for both candidates and recruiters.

Developed using Python, React.js, and open-source tools, the project adopts an agile methodology for iterative design and testing. By automating the recommendation, transcription, and evaluation processes, the Automated Interview System addresses the limitations of traditional and existing solutions, minimizing manual intervention while improving consistency and scalability. The project demonstrates a practical application of AI in recruitment, offering a cost-effective and reliable solution to streamline hiring workflows.

***Keywords**: Automated Interview System, AI, Recommendation, Transcription*

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NLP	Natural Language Processing
AI	Artificial Intelligence
TF-IDF	Term Frequency-Inverse Document Frequency
WAV	Waveform Audio File Format
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
DFD	Data Flow Diagram
JWT	JSON Web Token
URL	Universal Resource Locator
AMQP	Advance Message Queuing Protocol
ASR	Automatic Speech Recognition

CHAPTER 1

INTRODUCTION

1.1. Introduction.

The **Automated Interview System** is a web-based application designed to transform the interview process by automating the evaluation of technical candidates. The system focuses on enabling candidates to participate in interviews by answering questions verbally, with the questions recommended based on their details and expertise. Their responses are automatically transcribed and analyzed for similarity with a predefined answer set to ensure high accuracy and relevance. The results of the answer comparison are displayed after the completion of the interview.

Users choose the interview category, such as Frontend, Backend, or DevOps, and provide details about their expertise. Based on the user profiles, the system recommends questions using content-based filtering. The users then answer the recommended questions within a given time frame. The audio responses provided by the users are transcribed into text using the pre-trained Whisper AI model. These transcribed responses are compared with a predefined answer set using cosine similarity, and the results are displayed at the end of the interview process.

The Automated Interview System incorporates major algorithms to ensure efficiency and accuracy. A content-based filtering algorithm recommends interview questions aligned with the user's profile and expertise, utilizing a dataset that includes various topics. For audio-to-text conversion, the Whisper AI model accurately transcribes spoken responses into text. To evaluate the user's answers, a cosine similarity algorithm compares the transcribed responses against a predefined dataset. These algorithms work in harmony to provide a seamless and intelligent interview experience.

1.2. Problem statement

- Manual interview creation and evaluation required significant time, effort, and resources.
- Existing automated systems primarily focused on basic speech-to-text conversion.
- These systems failed to accurately tailor questions to candidates' expertise.

1.3. Objectives

- To develop a system that recommend relevant questions based on a candidate's expertise using content-based filtering, convert their speech to text, and display results by performing a cosine similarity operation with a predefined dataset.

1.4. Scope and Limitations

Scope:

- Automation of Interview Process: Streamlines the interview process by automating question recommendations, response transcription, and evaluation.
- Content-Based Question Recommendations: Tailors interview questions to candidates' expertise using content-based filtering techniques.
- Objective Evaluation: Compares responses against predefined datasets using cosine similarity to ensure unbiased and accurate assessments.
- Customizable Categories: Supports multiple interview categories such as Frontend, Backend, and DevOps, allowing flexibility in use.
- Real-Time Processing: Provides immediate results upon completion of the interview.
- Data Security**:** Ensures secure handling of sensitive user data through encryption and access control mechanisms.
- User-Friendly Interface: Simplifies navigation for both candidates and recruiters with an intuitive design.

Limitations:

- Dependency on Predefined Datasets: The quality of evaluation is dependent on the accuracy and completeness of predefined answer datasets.
- Limited Multilingual Support: Currently supports transcription and evaluation in a single language, restricting global applicability.
- Dependency on third party AI Model: Since speech to text translation is done by third party pre-trained model called Whisper AI, so if any fault occur on this model impact on the accuracy of translation.
- Hardware Requirements: Requires devices with microphones and stable internet connections for real-time audio processing.

1.5. Development Methodology

The project followed the structured methodology and follows agile approach. The Key stages include requirement gathering, design, implementation, and testing ensuring that the system meets user needs and adapts to challenges.

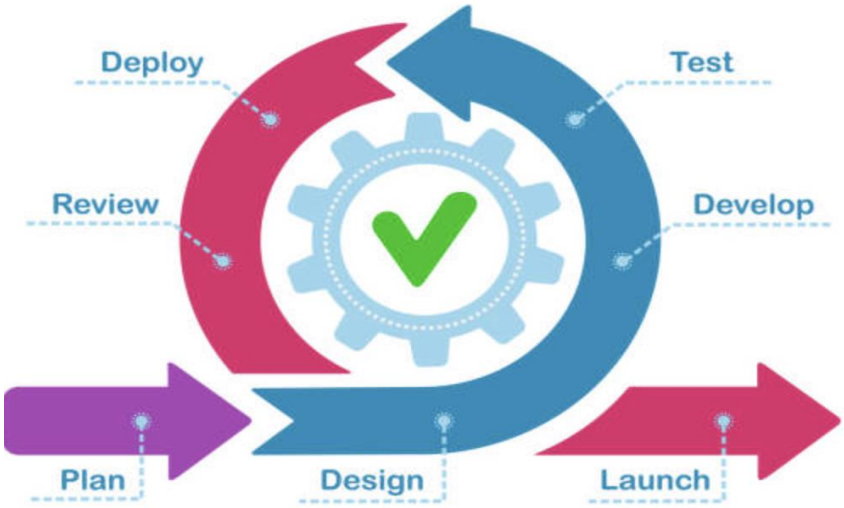

Figure 1: Agile Methodology

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

The project incorporates various modern technologies and algorithms to ensure efficient functionality and robust performance. This section provides an overview of the fundamental theories, general concepts, and terminologies relevant to the project.

JSON Web Tokens (JWT) are a compact, URL-safe method for representing claims between two parties. JWTs are widely used for authentication and secure data exchange in modern web applications due to their efficiency and simplicity. A JWT consists of three parts: a header, a payload, and a signature. The header specifies the type of token and the signing algorithm, the payload contains the claims (such as user information or permissions), and the signature ensures the token's integrity and authenticity. In this project, JWT is employed to securely manage user authentication by encoding and verifying user credentials. Its stateless nature and ability to embed essential information within the token itself enhance scalability and simplify the session management process [1].

RabbitMQ is a robust and widely used message broker that facilitates communication between distributed systems through message queuing. It supports advanced message queuing protocol (AMQP) and provides mechanisms for asynchronous communication.

- Producer: The sender of the message.
- Queue: Stores messages until they are processed.
- Consumer: The receiver that processes the message.

RabbitMQ ensures efficient handling of tasks and decouples the components of the system. In this project, RabbitMQ is implemented to queue operations such as processing user responses and generating reports asynchronously, ensuring scalability and responsiveness [2].

Cosine similarity, a concept rooted in vector mathematics and geometry, was developed to measure the similarity between two vectors by calculating the cosine of the angle between them. Initially introduced in fields such as information retrieval and computational linguistics, it has become a foundational technique in Natural Language Processing (NLP) and machine learning. Its primary advantage lies in its ability to determine the orientation

of vectors rather than their magnitude, making it particularly effective for high-dimensional datasets. Cosine similarity is widely used to compare text data by converting words or sentences into vector representations, enabling the assessment of semantic closeness. In this project, it is utilized to evaluate the similarity between the user's transcribed answers and predefined responses, ensuring a reliable and efficient method for analyzing the content's relevance and accuracy [3].

The Whisper AI model, developed by OpenAI, is a pre-trained model for automatic speech recognition (ASR) and speech translation. It is a Transformer-based encoder-decoder model, also known as a sequence-to-sequence model, trained on 680,000 hours of labeled speech data using large-scale weak supervision. Whisper models demonstrate exceptional generalization capabilities across various datasets and domains without requiring fine-tuning. Depending on the training data, the model supports both English-only speech recognition and multilingual tasks, including speech translation. For speech recognition, the model generates transcriptions in the same language as the audio, while for speech translation, it can provide transcriptions in a different language. In this project, the Whisper AI model was utilized to transcribe the user's verbal responses into text with high accuracy and efficiency, forming a critical component for the system's text analysis and comparison operations [4].

2.2. Literature Review

Several automated interview systems are currently available, each designed to streamline the recruitment process by leveraging technology to reduce human intervention. These systems typically focus on different aspects of the interview process, such as video interviews, AI-driven assessments, and speech-to-text conversions. The major are:

Modern Hire was developed by Shaker International, a highly advanced hiring platform designed to streamline the recruitment process for organizations looking to make data-driven hiring decisions. It integrates innovative features such as video interviews with AI-driven assessments, allowing a comprehensive evaluation of candidates by analyzing both verbal and non-verbal cues. The platform was created to enhance recruitment efficiency by using predictive analytics to assess candidate potential and fit for specific roles. Additionally, Modern Hire offers automated scheduling and AI-powered scoring, reducing manual efforts and ensuring consistent, objective evaluations. However, despite its strengths, it lacks a robust question recommendation system that can tailor questions based

on a candidate's expertise, limiting its adaptability in customizing interviews to individual profiles. This gap underscores the need for a more dynamic system that prioritizes personalized question recommendations to enhance the overall interview process [5].

Interviewer.AI mainly concerned about efficiency. It's a state-of-the-art video recruiting software that leverages Generative AI and Explainable AI to automate job descriptions, craft relevant interview questions, and pre-screen and shortlist candidates. This ensures candidate find the best talent for their roles while maintaining content relevancy. Advanced online smart video interview software significantly reduces the time spent on unnecessary pre-interviews, allowing to focus on what truly matters [6] .

iMocha offers more than 2,000 skills assessments that have been benchmarked by industry experts and built with AI-powered proctoring measures for thorough cheating prevention. They also provide video-based questions and face-to-face interviewing and allows to hire collaboratively with other hiring managers [7].

Brazen offers virtual hiring events to allow recruiters and hiring managers to engage with multiple candidates during a specified time, all in one place. User can customize the event with your branding, give candidates resources about your organization, broadcast live video to all attendees, create separate event booths per open position, and connect with each candidate one-on-one [8].

HireVue offers conversational AI, video interviewing, skills assessments, and automated interview scheduling all in one place. Built with ATS integration, this platform provides structured interviews with questions, templates, and evaluation guides for jobs at all levels, as well as live video interview software [9].

CHAPTER 3

SYSTEM ANALYSIS

3.1. System Analysis

3.1.1. Requirement Analysis

i) Functional Requirements

- User Authentication**:** The system allowed users to create accounts, log in securely, and manage their account.
- Question Recommendation**:** Based on the users' selected categories, the system recommended relevant questions using a content-based filtering algorithm.
- Real-Time Audio Recording and Transcription: The system recorded users' responses in real-time and transcribed the audio responses into text using the pre-trained Whisper AI model.
- Answer Comparison and Displaying Results: The system compared the transcribed text responses with predefined answers using cosine similarity. It calculated the similarity scores and generated detailed comparison reports.

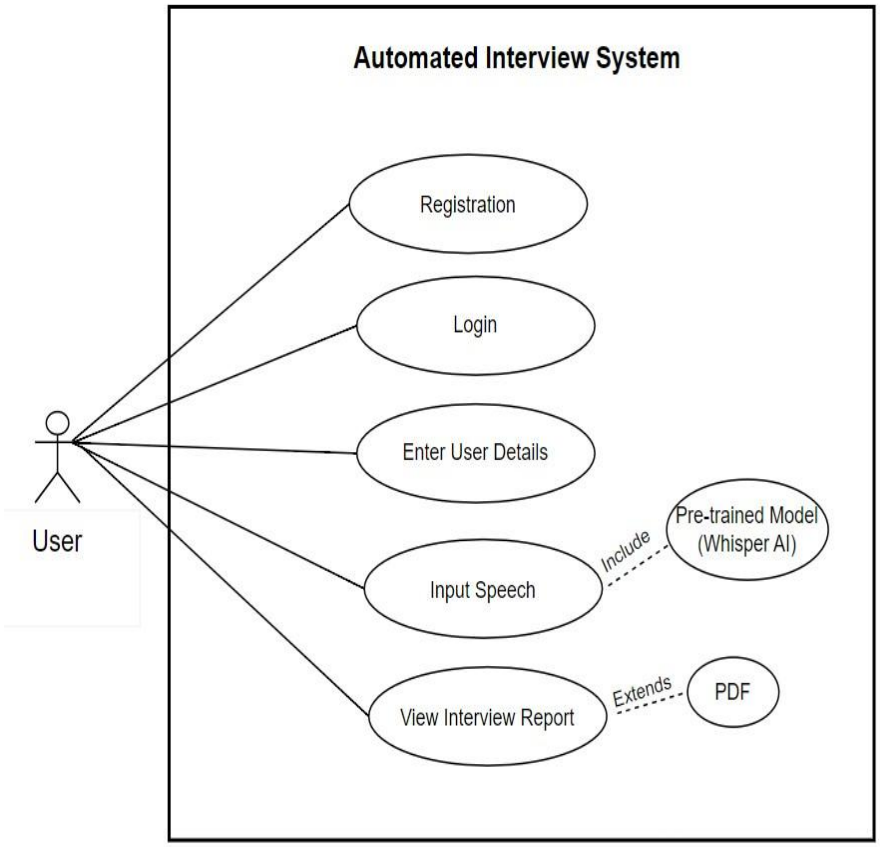

Figure 2: Use case diagram of Automated Interview System

ii) Non-Functional Requirements

- Security: The system ensured the security and confidentiality of user data, including personal information and interview responses. Robust authentication mechanisms, data encryption, and secure storage practices were implemented to protect sensitive information.
- Usability: The user interface was designed to be intuitive and user-friendly, allowing users to navigate the system with ease. It supported clear instructions, straightforward interactions, and a responsive design to cater to various devices.
- Maintainability**:** The system was developed for ease of maintenance, featuring clear and well-documented code, modular components, and automated testing. Regular updates and bug fixes were manageable with minimal disruption to the system's operation.
- Reliability: The system demonstrated high reliability with minimal downtime or failure rates. It was robust against errors and capable of recovering gracefully from unexpected issues or system crashes.

3.1.2. Feasibility Study

- **Economic Feasibility:** The project emphasized a cost-effective approach while delivering substantial benefits. By utilizing open-source technologies such as Python and React.js, alongside the Whisper AI model for speech-to-text conversion, development costs were minimized. Ongoing maintenance expenses were managed through routine updates and efficient automated processes, reducing long-term financial commitments. The system’s capabilities in accurately recommending questions and evaluating responses enhanced the efficiency of the interview process, resulting in potential recruitment cost savings and operational improvements.
- **Operational Feasibility:** The system was operationally feasible, designed to be user-friendly and straightforward. By automating the majority of its operations in sequence, the system allowed recruiters and HR professionals to manage interviews and assess candidate responses with ease. Complex tasks such as question recommendation, audio transcription, and text comparison were automated, minimizing the need for manual intervention. Integration with existing HR and recruitment platforms was streamlined, ensuring seamless incorporation into current workflows.

- **Technical Feasibility:** The system was technically feasible due to the availability of advanced, well-documented technologies and pre-trained models such as Whisper AI for accurate audio-to-text conversion and content-based filtering algorithms for question recommendations. Widely supported frameworks like React.js and Python ensured robust and scalable development, while open-source tools and libraries facilitated cost-effective and efficient implementation.
- **Schedule Feasibility:**

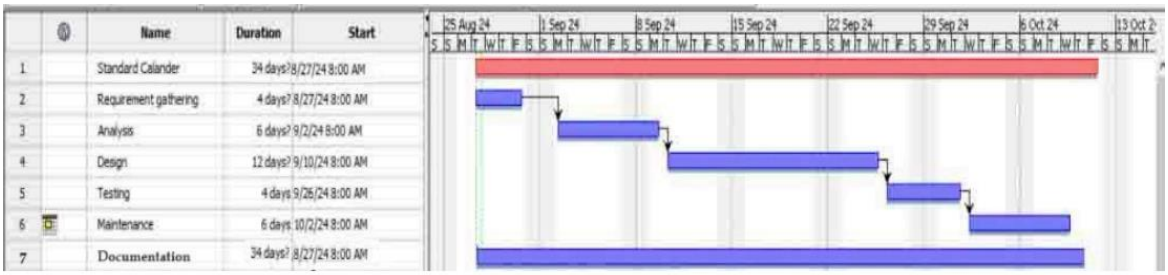

Figure 3: Gantt Chart

3.1.3. Analysis
Structured Approach:

The structured approach refers to a methodology in which the system's analysis, design, and development processes are carried out using well-defined and organized models and diagrams. It focuses on processes, data flow, and system components in a hierarchical manner. For your project, adopting a structured approach involves using diagrams like ER diagrams, DFDs, and clearly defining the flow of information.

- Data modeling using ER Diagrams

Use an ER Diagram to represent the relationships between entities such as users, questions, answers, and categories.

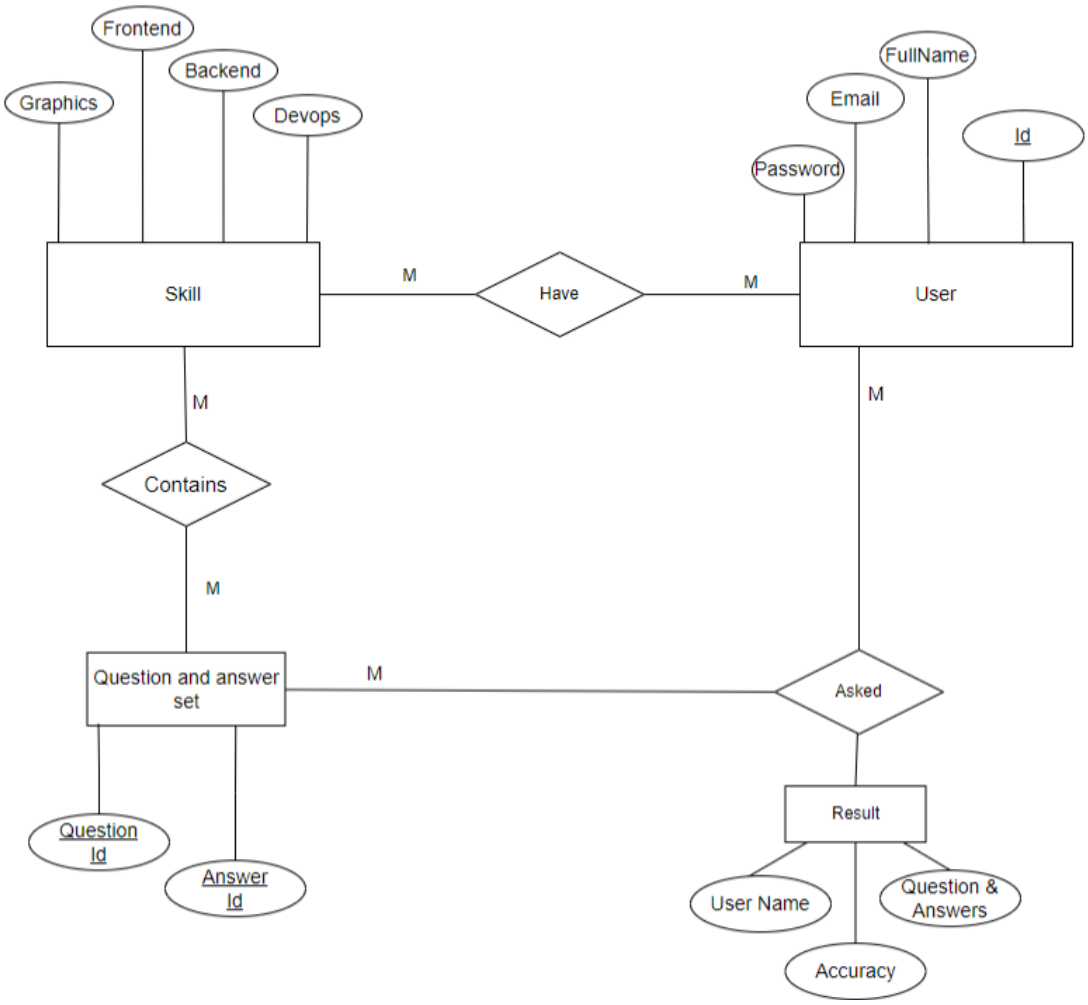

Figure 4: ER Diagram of Automated Interview System

The above ER diagram shows the relationship between the User, Skill, and Question & Answer set and Result of the interview process. The Entities are represented as rectangle and attributes are represented in oval shape. The entity have their corresponding attributes and the diamond shape represent the relationship between them.

- Process modeling using DFD

Use Data Flow Diagrams (DFDs) to represent the data flow between system components:

Level 0 DFD: Illustrates the overall system, showing data flowing between the user, authentication module, question recommendation system, and report generation.

Level 1 DFD: Breaks down each major component (e.g., the question recommendation system) into smaller processes.

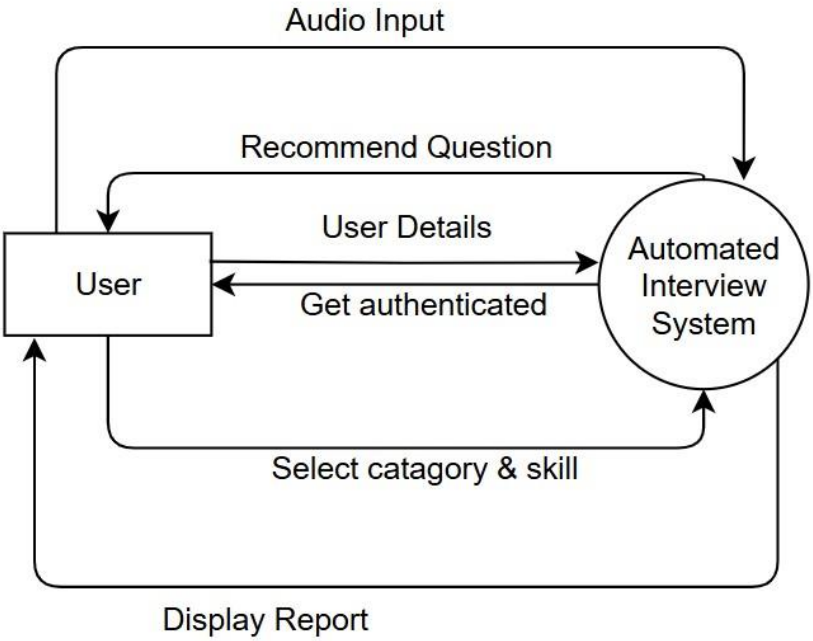

Figure 5: DFD level 0 of Automated Interview System

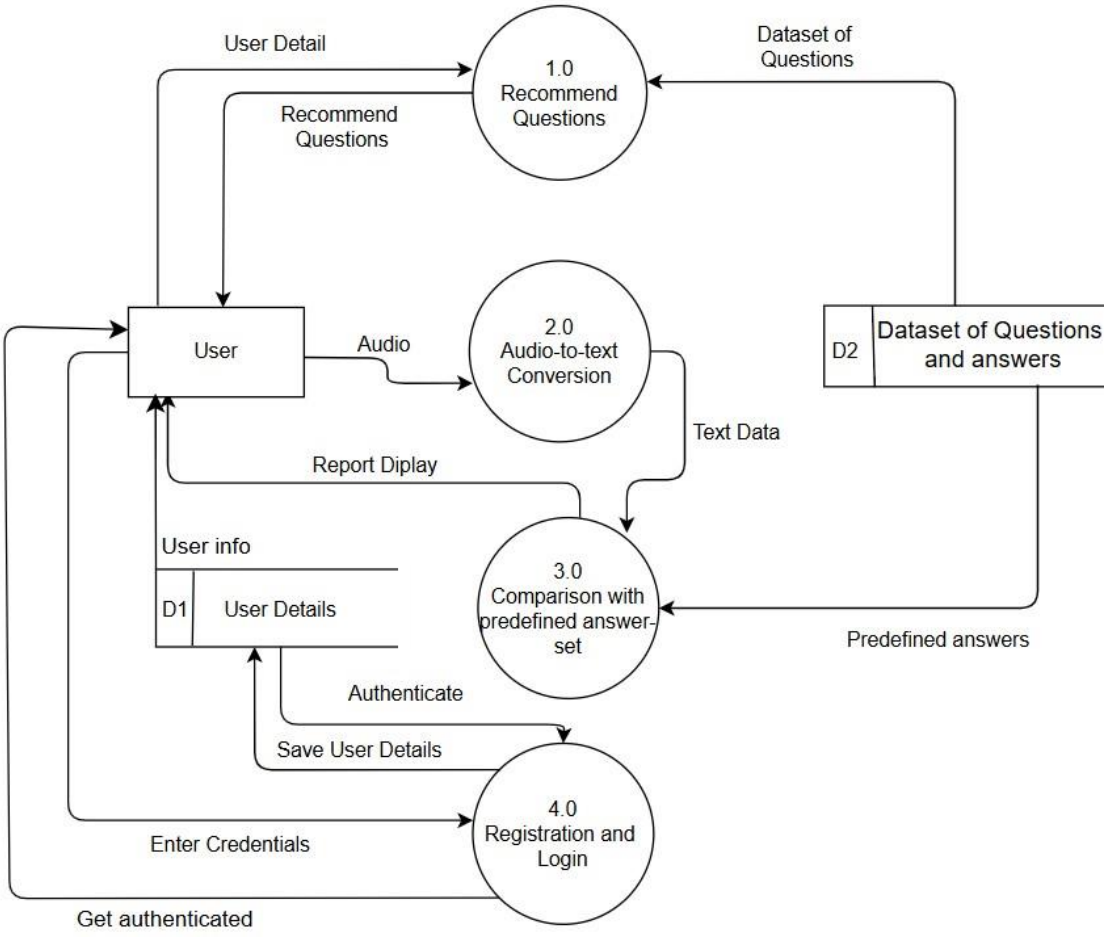

Figure 6: DFD level 1 of Automated Interview System

CHAPTER 4

SYSTEM DESIGN

4.1. Design

Structured Approach:

Database Design:

The database schema implemented is NoSQL, so it contains the data in format of JSON. The user details at the time of user-registration is stored as:

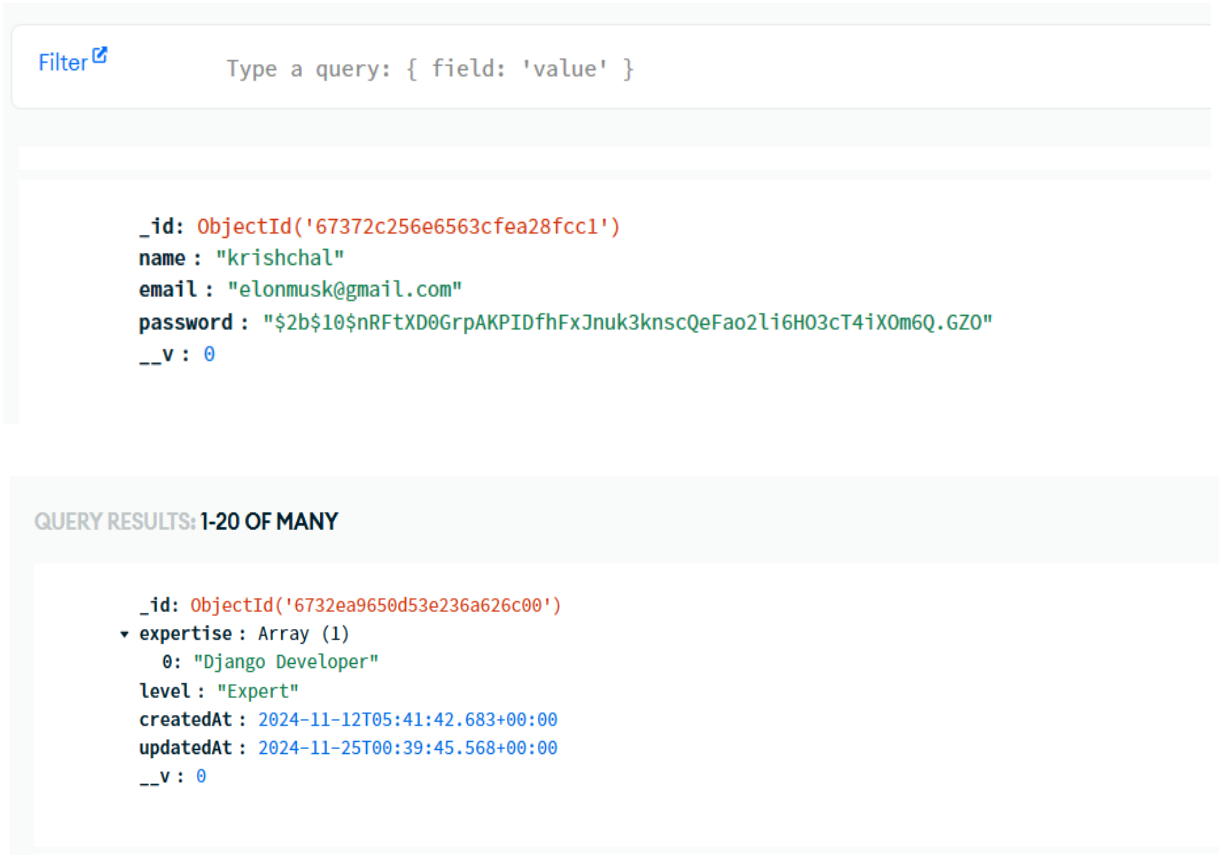

Figure 8: Database Structure of Automated Interview System

4.2. Algorithms Details and Flowchart

Content Based Recommendation Algorithm

The Question Recommendation Algorithm is designed to match interview questions to a candidate's selected expertise level and chosen technology. This algorithm implemented the content-based techniques to ensure that the questions posed to a candidate are relevant to their skills and experience. The goal is to enhance the interview process by tailoring it to the individual, thereby improving the accuracy of the assessment.

The algorithm approach is as follows:

1) Preprocessing: The algorithm first preprocess the user's selected category related keywords and the dataset of questions. This involve tokenizing and normalizing text data from both sources to prepare it for analysis. Specifically, the text is converted to lowercase, and stopwords and punctuations are removed to focus on significant keywords.

2) Feature Extraction: Using the Term Frequency-Inverse Document Frequency method, the algorithm convert the preprocessed text data into numerical vectors. This transformation capture the importance of each word in the context of the entire dataset and user category of technology.

Term Frequency (TF): Measures how frequently a term't' which is users expertise related keywords present in document and 'd' represent the total number of words present in document of the questions and answer. Mathematically expressed as:

TF (t, d) = total number of times t appear in total number of terms in d

Inverse Document Frequency (IDF) : Measures the importance of the term across all documents where, 't' represent the term or word based on we are recommending questions , 'N' represent the total number of document present in project corpus and DF(t) represent the number of document that contains the term 't' .

IDF (t) = log NDF (t)

*TF IDF (t) = TF (t, d) * IDF (t)*

3) Similarity Calculation: The algorithm compute the similarity between the user's expertise and the available questions using cosine similarity. This achieved by calculating the dot product of the vectors of the user's expertise and each question, and then dividing by the product of their magnitudes. Cosine similarity determine how closely a question matches the user's expertise.

$$\text{Cosine Similarity} = \cos\Theta = \frac{A.B}{|A||B|} = \frac{\sum_i^n A\,B}{\sqrt{\sum_i^n A^2}\sqrt{\sum_i^n B^2}}$$

Whereas 'A' represents the user's expertise term vector and 'B' represents the vector for the questions term present in the document. $\cos\Theta$ represent the angle between both vectors, less the angle means more suitable for recommendation.

4) Recommendation Generation: Based on the calculated similarities, the algorithm rank the questions. It selects and recommends the top questions that have the highest similarity scores to ensure relevance and appropriateness for the user's level and expertise.

Comparison Algorithm

The answer given by the user and the best possible answer present in the dataset is compared in order to calculate the result. The answer given by user and pre-defined answer both are converted into vector form and following steps proceeds:

1. Preprocessing Text:

The text data is prepared for similarity comparison by tokenizing and removing irrelevant characters.

Tokenization: The input text is splited into individual words (tokens).

Lowercasing: All tokens are converted to lowercase to ensure uniformity.

Removing Punctuation and Stop Words: Common stop words and punctuation marks are filtered out to focus on meaningful content

2. Calculating Cosine Similarity :

The similarity between users' answer and dataset set answer are measured based on their content.

Create Token Vectors: The preprocessed text are converted into vectors based on word counts. Each vector are represents the frequency of unique words in the text.

Dot Product: The dot product of the two vectors are computed to measure the degree of similarity.

Magnitude Calculation: The magnitude of each vector are calculated to normalize the similarity measure.

Cosine Similarity = $\cos\Theta = \frac{A.B}{|A||B|} = \frac{\sum_i^n A\ B}{\sqrt{\sum_i^n A^2}\sqrt{\sum_i^n B^2}}$

3. Comparison Result: Computed best possible result based on the answer text comparison.

Major Steps Performed

- **Calculate Cosine Similarity**: The manual calculation function was implemented to determine the similarity score between predefined and input text.
- **Repeated Words**: The occurrences of words appearing more than once in the input text were counted.
- **Precision**: The proportion of words in the input text that matched those in the predefined text was measured.
- **Accuracy**: Cosine similarity was converted into a percentage to represent overall accuracy.

Speech-to-Text Conversion Algorithm

The **Whisper AI model** was utilized to transcribe spoken language into text with high accuracy. This algorithm leveraged deep learning techniques to analyze audio data and transform it into coherent text. Unlike traditional speech recognition systems, Whisper handled a variety of languages and accents, making it highly versatile.

The implemented algorithm followed these steps:

1. **Audio Recording**: Audio was recorded from the microphone for a specified duration.
2. **Store the Recorded Audio**: The recorded audio data was stored in an array format and saved as an audio file.
3. **Convert to WAV File**: The recorded audio data was converted into a WAV file using the scipy.io.wavfile module.
4. **Model Initialization**: The pre-trained Whisper AI model (‘medium’) was loaded into the system.
5. **Transcribe Audio**: The Whisper model transcribed the audio from the WAV file into text.
6. **Save Transcription**: The transcription output was stored for further processing and analysis.
7. **Delete Temporary File**: The temporary WAV file was deleted to free up resources and maintain system efficiency.

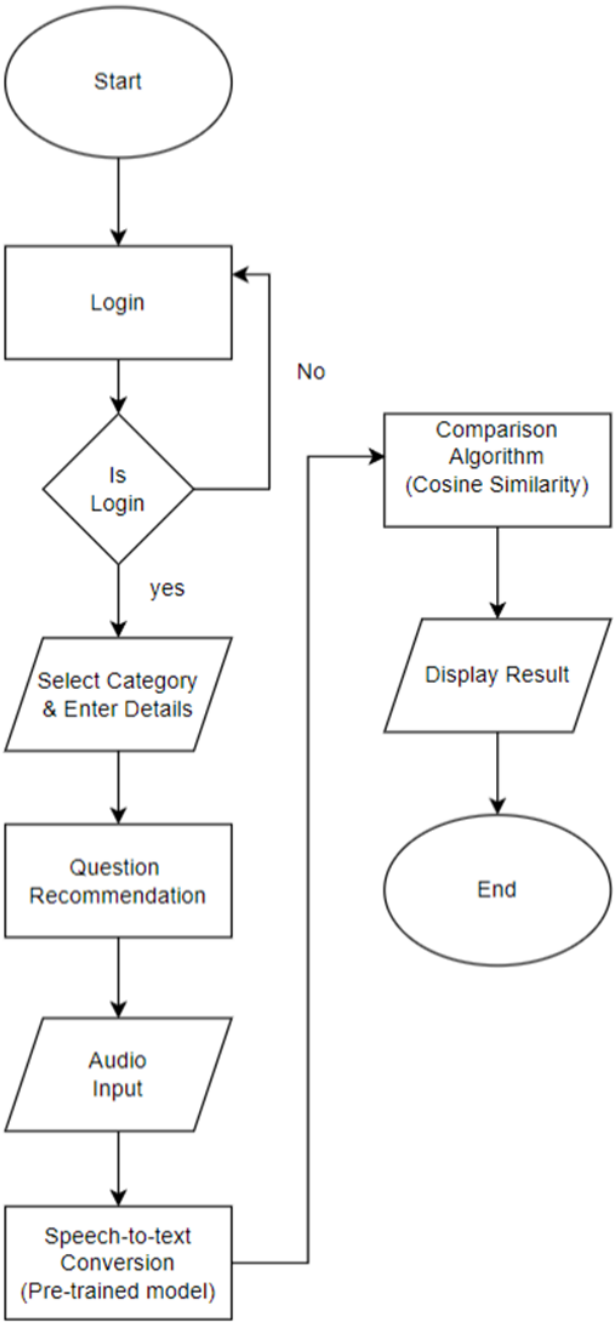

Figure 7 : Flow-Chart Diagram of Automated Interview System

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1. Tools Used

- Draw.io**:** Draw.io was used to create and design required diagrams for the documentation purposes.
- GitHub: GitHub was used as version control tool and repository sharing among team members for reviewing and making changes.
- Visual Studio Code**:** Visual Studio Code was used as platform for coding due to its versatility and flexibility.
- Project Libre : Project Libre was used to draw Gantt charts for and monitor work progress.

Languages Used:

- Python: Python was selected as the primary programming language due to its versatility with Flask as backend framework and extensive libraries imported are:
  Pandas: It was used for handling structured data efficiently, allowing seamless manipulation, filtering, and organization of textual datasets. Its Data Frame functionality simplifies data analysis and transformation during preprocessing and report generation.
  NumPy: It was utilized for numerical computations, including the creation and manipulation of TF-IDF vectors. Its optimized array operations enable efficient cosine similarity calculations for text analysis.
  NLTK Corpus : It provided access to extensive linguistic resources like stopwords, which are essential for text preprocessing. These resources help remove irrelevant words and ensure accurate tokenization and analysis of input data.
- HTML: HTML was utilized to structure the content of the web application, including the creation of essential elements like headings, paragraphs, forms, and buttons. It defined the layout and semantic organization of the user interface, ensuring that the application's content was well-structured and accessible.
- CSS**:** CSS was employed to style and design the web application's appearance. It controlled visual aspects such as colors, fonts, spacing, and layout, enhancing the

overall user experience by ensuring that the application was aesthetically pleasing and responsive across various devices and screen sizes.

- JavaScript: JavaScript was used to implement dynamic and interactive features within the web application. The JavaScript library React.js was utilized to create a single-page frontend architecture and Node.js for server side language. React handled client-side logic such as form validation, dynamic content updates, and user interactions, contributing to a more engaging and functional user interface.

5.2 Testing

5.2.1. Test Cases for Unit Testing

Individual units or module of the system are tested properly, by which following result is occurred.

a) For Sign up

Test Scenario	Test Steps	Input Test Data	Expected Result	Test Status
Check signup activity with valid data	**1.** Open App, 2. go to signup activity 3. Fill up form with valid data and click to signup	Name: Krishchal Regmi Email: krish@gmail.com Password: testing@123	User should get success message and redirect to login page	PASS

b) For Login

Test Scenario	Test Steps	Input Test Data	Expected Result	Test Status
Check login activity with valid data	1. Open App, 2. go to login activity	Email: krish@gmail.com	User should get success message and	PASS

	3. Fill up form with valid data and click to login	Password: testing@123	redirect to dashboard	

c) For question recommendation

Test Scenario	Test Steps	Input Test Data	Expected Result	Test Status
displaying question on users screen	1. Open App, 2. select category 3. choose expertise 4. Click next	check boxes on screen	User should get displayed the recommended questions	PASS

d) For audio streaming and transcribing

Test Scenario	Test Steps	Input Test Data	Expected Result	Test Status
Give microphone access and start speaking	1. Click on start button 2. start speaking answer 3. Click next	Users speech answer	transcribe result on server side console	PASS

e) For Cosine Similarity Comparison Result

Test Scenario	Test Steps	Input Test Data	Expected Result	Test Status

Displaying cosine similarity score at the result section of dashboard	After completion of interview 1. go to Reports section on dashboard and observe.	As expected	The bar and chart of score.	PASS

5.2.2. System Testing

The system testing has been performed by testing whole application in distributed architecture, with the connection of internet. The whole system runs in two devices, which includes Flask server with Whisper AI in one device and another device runs the server side JavaScript and frontend. Since, the database and the RabbitMQ requires internet connection, the system cannot run without internet.

System includes successfully executing these major modules:

i) Registration and Login
ii) Selection of category and Question Recommendation
iii) Audio transcribing into text and send in queue
iv) Performing Cosine Similarity for answer comparison

5.3. Result Analysis

The overall test analysis for the project demonstrates successful functionality across all key modules. The Sign-up and Login activities passed with valid data, correctly redirecting users to the login page and dashboard, respectively. The question recommendation system successfully displayed recommended questions based on the selected category and expertise. The audio streaming and transcription module accurately transcribed spoken answers on the server-side console. Finally, the cosine similarity comparison effectively displayed scores as bar and chart visualizations in the report section of the dashboard. All tests met the expected results, confirming a functional and reliable system.

Input:

Category selected: Backend

Skills selected: Node.js, API, Asynchronous

Intermediate Processing and Results of Question Recommendation:

Category selection input tokens : ['backend', 'node.js', 'api', 'asynchronous']

It is already pre-processed .

Most relevant question based on this input token is selected by performing the TF-IDF with each questions.

Vectorization :

TF Vector (Inputs Tokens):

[1/4, 1/4, 1/4, 1/4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] = [0.25, 0.25, 0.25, 0.25, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1) Question 1: What is middleware in Express.js, and how is it used?

keywords with this question is : ['middleware', 'Node.js, 'API', 'Backend']

TF Vector of this particular question: [0, 0, 0, 0, 1/3, 1/3, 1/3, 0, 0, 0, 0, 0, 0, 0, 0, 0] = [0, 0, 0, 0, 0.33, 0.33, 0.33, 0, 0, 0, 0, 0, 0, 0, 0, 0]

IDF Values:

Word	Backend	Node.js	API	Asynchronous	Middleware	Express.js
Doc Count	2	2	2	1	1	1
IDF	1.40	1.40	1.40	1.79	1.79	1.79

TF-IDF Vector (Inputs):

[0.25 * 1.79, 0.25 * 1.40, 0.25 * 1.40, 0.25 * 1.40, 0, ..., 0] = [0.447, 0.35, 0.35, ..., 0]

TF-IDF Vectors (Questions): Similar calculation is applied to all questions.

Cosine similarity is calculated between the TF-IDF vector of the inputs and each question and the highest question is sent through API.

Recommended Question :

2) What is middleware in Express.js, and how is it used?

User's speech transcribed answer: Middleware refers to functions that execute during the lifecycle of a request to the server. Each middleware function has access to the request object, response object, and a next function, which allows it to pass control to the next middleware in the stack. Middleware can perform tasks such as processing requests, modifying responses, and handling errors.

Predefined answer: Middleware in Express.js refers to functions executed during the lifecycle of a request-response cycle. They have access to the request (req) and response (res) objects and can modify them, execute additional code, or terminate the request-response cycle. Middleware is commonly used for authentication, logging, error handling, or modifying request data. For example, middleware can check if a user is authenticated before accessing certain routes or log details of each incoming request. Express allows

Intermediate Processing and Results of Answer Comparision:

User Transcribed Answer Tokenized as :

["middleware", "refers", "to", "functions", "that", "execute", "during", "the", "lifecycle", "of", "a", "request", "to", "the", "server", "each", "middleware", "function", "has", "access", "to", "the", "request", "object", "response", "object", "and", "a", "next", "function", "which", "allows", "it", "to", "pass", "control", "to", "the", "next", "middleware", "in", "the", "stack", "middleware", "can", "perform", "tasks", "such", "as", "processing", "requests", "modifying", "responses", "and", "handling", "errors"]

Pre-defined Answer Tokenized as :

["middleware", "in", "express.js", "refers", "to", "functions", "executed", "during", "the", "lifecycle", "of", "a", "request-response", "cycle", "they", "have", "access", "to", "the", "request", "req", "and", "response", "res", "objects", "and", "can", "modify", "them", "execute", "additional", "code", "or", "terminate", "the", "request-response", "cycle", "middleware", "is", "commonly", "used", "for", "authentication", "logging", "error", "handling", "or", "modifying", "request", "data", "for", "example", "middleware", "can", "check", "if", "a", "user", "is", "authenticated", "before", "accessing", "certain", "routes", "or", "log", "details", "of", "each", "incoming", "request", "express", "allows", "chaining",

"multiple", "middleware", "functions", "providing", "flexibility", "in", "application", "behavior"]

Preprocessed tokens (Stopwords and Puntuations Removed):

Users-Transcribed Answer :

["middleware", "refers", "to", "functions", "that", "execute", "during", "the", "lifecycle", "of", "request", "server", "each", "has", "access", "object", "response", "next", "which", "allows", "it", "pass", "control", "in", "stack", "can", "perform", "tasks", "such", "as", "processing", "modifying", "and", "handling", "errors"]

Pre-defined Answer :

["middleware", "in", "express.js", "refers", "to", "functions", "executed", "during", "the", "lifecycle", "of", "request-response", "cycle", "they", "have", "access", "to", "req", "res", "objects", "and", "modify", "logging", "authentication", "code", "terminate", "authentication", "example", "check", "user", "authenticated", "before", "accessing", "certain", "routes", "log", "details", "incoming", "express", "allows", "chaining", "multiple", "flexibility", "application", "behavior"]

Vectorization :

User’s Transcribed Answers Vector:

[5, 1, 4, 2, 1, 1, 7, 1, 1, 2, 1, 2, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 0, 0, 0]

Pre-defined Answers Vector :
[7, 1, 4, 2, 1, 1, 8, 1, 1, 3, 3, 2, 0, 1, 1, 1, 4, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 0, 0, 2, 0, 1, 1, 1, 1]

Now vector A represented by Users transcribed answer and vector B represented by pre-defined answers and cosine similarity is performed between them

Cosine Similarity has performed between vector A and B which given the result : Cosine Similarity Score : 0.71

CHAPTER 6

CONCLUSION AND FUTURE RECOMMENDATIONS

6.1. Conclusion

The Automated Interview System effectively fulfills its objective of enhancing the interview process through advanced technological integrations. By employing content-based filtering, the system recommends the most relevant and tailored questions aligned with the user's expertise, ensuring a personalized and efficient interview experience. The integration of the Whisper AI model ensures accurate and reliable speech-to-text conversion, significantly reducing errors in capturing user responses. Additionally, the cosine similarity algorithm provides robust and dependable evaluations by comparing candidate responses with a predefined answer set, ensuring precise and fair assessments. All specified requirements and project objectives, including user authentication, dynamic question recommendations, accurate answer evaluations, and comprehensive report generation, have been successfully implemented and achieved, making the system a comprehensive solution for automated interviews.

6.2. Future Recommendations

While the Automated Interview System effectively streamlines the interview process and provides accurate assessments, there are several areas for future enhancement to make the system more robust and versatile. The following recommendations can be considered for further development:

- Multilingual Support
- Inbuilt Resume-Scanner

And by integrating more advance AI feature will expand the scope of the system to make recruitment process easier.

.

REFERENCES

[1] **M. B. Jones**, "JSON Web Token," *[Online]*. Available: *https://jwt.io/introduction.* [Accessed **04 12 2024**].

[2] "**rabbitMQ**," VMware, *[Online]*. Available: https://www.rabbitmq.com/. [Accessed **04 12 2024**].

[3] Y. Januzaj and A. Luma, "researchgate," 11 2022. *[Online]*. Available: *https://www.researchgate.net/publication/361451105_Cosine_Similarity_.* [Accessed **04 12 2024**].

[4] "**Wishper**," OpenAI, *[Online]*. Available: *https://openai.com/index/whisper/.* [Accessed **04 12 2024**].

[5] **S. International**, "Modern Hire," *[Online]*. Available: *https://modernhire.com/wp-content/uploads/simple-file-list/Modern-Hire-Technology-Overview.pdf*. [Accessed **23 11 2024**].

[6] **S. Saurabh**, "Interview.AI," *[Online]*. Available: *https://interviewer.ai*. [Accessed **23 11 2024**].

[7] **S. K. Amit D Mishra**, "iMocha," [*Online*]. Available: https://imocha.io. [Accessed **01 12 2024**].

[8] **M. Abbey**, "Brazen," *[Online]*. Available: *https://brazen.com.* [Accessed **03 12 2024**].

[9] **M. Newman**, "HireVue," *[Online]*. Available: *https://hirevue.com*. [Accessed **04 12 2024**].