AudioMitra

An Innovative Content Transformation Platform

Let's look at our application now......

Audio Mitra

localhost:3000

Import bookmarks... | Getting Started | Apply1 minute | GitHub - romkatv/pow... | GitHub - ohmyzsh/oh... | 6.172 Performance Eng... | New Tab | Course: Software Engi... | Data Foundation | Audio Mitra | Data Foundation | Online Video Downloa... | Any Prompt Generator | Prompt Engineering G... | Microsoft Teams classic | Strivers A2Z DSA Cours...

AUDIO
मित्र

Home   Converter   About Us   Contact Us   Login

Audio Mitra Powered by SE Team_11

Pioneering Innovation through

Unified Solutions

Features

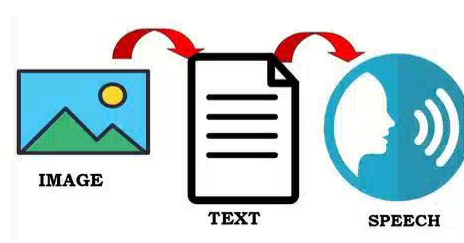### Image To Text
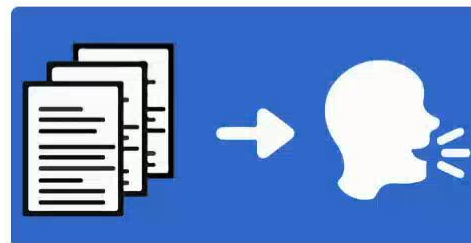
Text To Speech

Image To Text To Speech

Meet Our Team

Vilal Ali Zaidi
Senior Software Engineer (Full Stack)

Shriom Tyagi
Senior Software Engineer (Full Stack)

Madan NS
Senior Software Engineer (Full Stack)

Hanuma Bhaskara
Senior Software Engineer (Full Stack)

Introduction

AudioMitra is a Content Transformation Platform aimed at providing authentic and trustworthy transformations using the computational power of artificial intelligence and the cognitive power of humans in the loop.

Problem Statement

Audiomitra stands as an innovative solution within the realm of language processing, aiming to streamline human endeavors through technological advancement. Functioning as a content transformation system, Audiomitra facilitates the transition from digital images into text through Optical Character Recognition (OCR), and then converting text into audio using a text-to-speech (TTS) system.

Abstract Overview of the Proposed Solution

Text Extraction (OCR):

Open-Source API Utilization: Extract text content from images using the Open-Source OCR APIs.

Flexibility: Adaptable to other open-source OCR APIs or tesseract engine.

Output: Text files, with optional reassembly to match original document layout.

Content Validation:

Text Accuracy: Validate OCR'd content for text accuracy.

Human Intervention: Optionally allow for text editing and refinement.

Output: Validated text content.

Text to Speech/Audio (TTS):

Open-Source API Utilization: Convert extracted/validated content into speech/audio.

Flexibility: Adaptable to other open-source TTS APIs.

Output: Ready Audio file for play.

System Requirements

- **Functional Requirements (FR1-FR10):**
  - FR1: Image to Text conversion: Convert Images into Text using OCR.
  - FR2: Content validation and editing: Validate and edit text content for accuracy.
  - FR3: Text to audio conversion: Convert text content into audio formats using TTS.
  - FR4: Multilingual audio support: Support multiple languages in audio conversion.
  - FR5: User management: Support multi-user access with different roles.
  - FR6: Authentication and authorization: Secure user access to the system.
  - FR7: Human in the Loop: Enable collaboration between human and machine intervention.
  - FR8: User-friendly interface: Provide a web interface for accessing the system.
  - FR9: API integration: Integrate with open-source OCR and TTS APIs.
  - FR10: Flexibility with APIs: Adapt to different OCR and TTS APIs.

System Requirements

- **Non-Functional Requirements (NFR1-NFR7):**
  - NFR1: Performance and scalability: Handle large volumes of data efficiently.
  - NFR2: High availability: Ensure minimal downtime and fault tolerance.
  - NFR3: Reliability: Provide consistent and dependable service.
  - NFR4: Security: Use encryption and access controls for user data.
  - NFR5: Comprehensive documentation: Provide user guides and help materials.
  - NFR6: Standardized APIs: Use APIs for integration with third-party applications.
  - NFR7: Modular architecture: Facilitate the addition of new features.

Who will use our application

Stakeholders

Audiomitra Stakeholders

Users and Students	Libraries and Archives	OCR APIs Provider
Educat Institutions	Authors and Writers	TTS APIs Provider
Publishing Houses	Content Creators	Podcast Provider

Legend
person

Concerns

Audiomitra Concerns

Users and Students
General Users and Students

Libraries and Archives
Convert text into Audio

OCR APIs Provider
Open Source APIs Provider

Educat Institutions
Educational materials in multiple formats

Authors and Writers
Who create content in various formats

TTS APIs Provider
Open Source APIs Provider

Publishing Houses
Companies that produce text into multiple languages
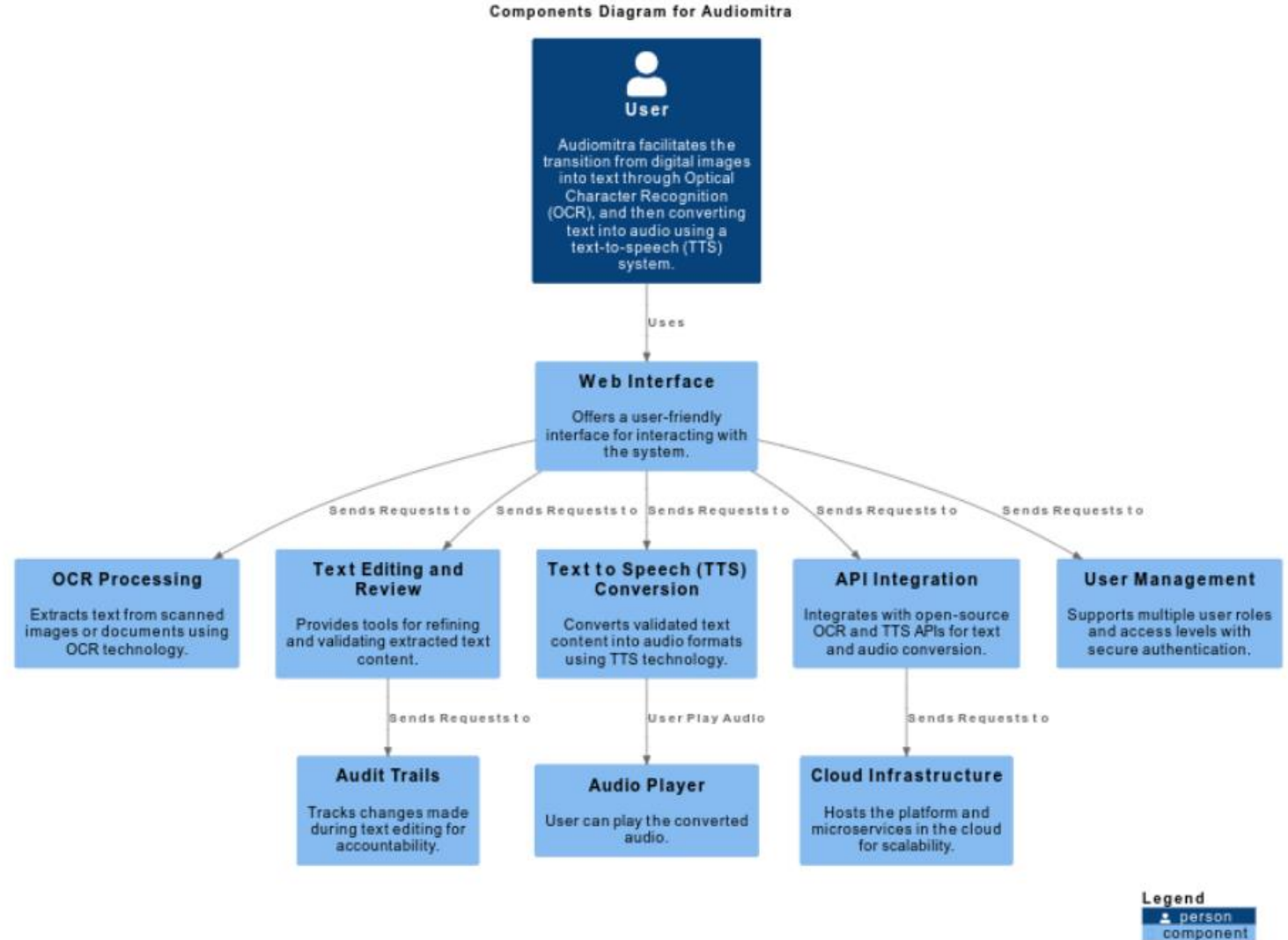
Content Creators
Who create content in various formats

Podcast Provider
Podcast organization create content in various formats

Legend
person

Components Diagram

Components Diagram for Audiomitra

Architectural Design Decisions Overview

Ensuring Robustness, Scalability, and Maintainability

Well-documented APIs: Interoperability with external systems for enhanced functionality.

Modern frontend: ReactJS + Tailwind CSS for utility-first, highly customizable designs.
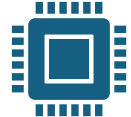
Lightweight backend: Python Flask for rapid development and RESTful APIs.

Reliable data storage: MySQL for structured data with transaction support and optimization.

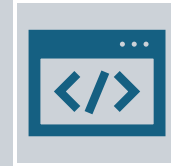Containerization: Docker + Kubernetes for consistency, scalability, and microservices deployment.

Microservices Architecture: Scalability, flexibility, and effective integration of diverse technologies.

Frontend and Styling Decisions

Utilized ReactJS with Tailwind CSS for frontend development.

Tailwind CSS's utility-first approach accelerates customization without dense style sheets.

Well-documented APIs designed for seamless integration with external systems like weather forecasts and ticketing platforms.
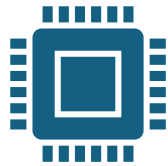
Interoperability enhances functionality, accuracy, and reliability of AudioMitra.

Backend, Database, and Architecture

Python Flask chosen for backend development due to its lightweight and flexible nature, ideal for RESTful APIs.

MySQL selected for database storage, offering reliability, transaction support, security, and performance optimization.
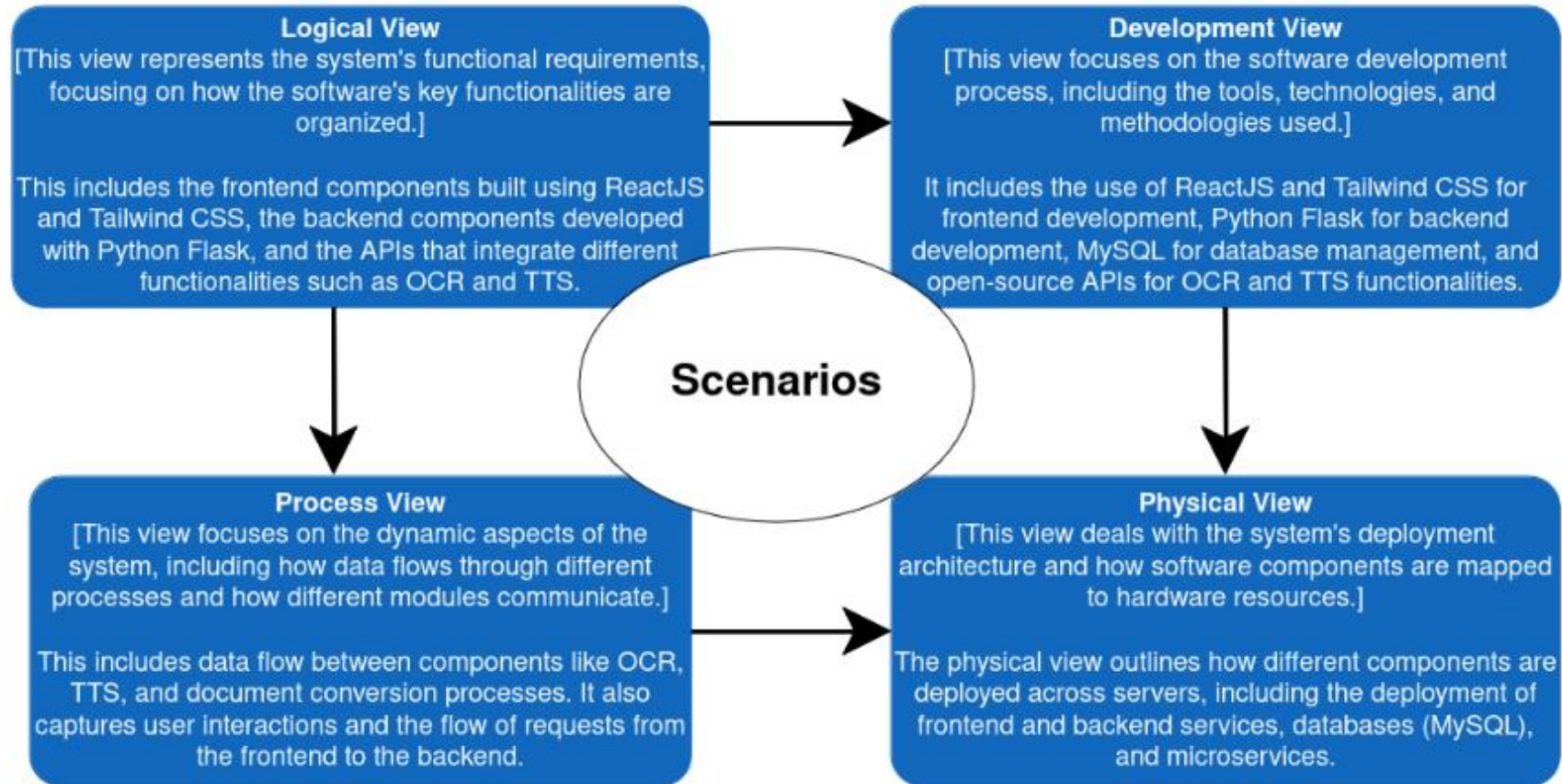
Adoption of Docker and Kubernetes for frontend and backend deployment ensures consistency, scalability, and automation.

Microservices Architecture implemented for scalability, flexibility, independent development, effective load balancing, and integration of diverse technologies tailored to specific services.

Views

Logical View
[This view represents the system's functional requirements, focusing on how the software's key functionalities are organized.]

This includes the frontend components built using ReactJS and Tailwind CSS, the backend components developed with Python Flask, and the APIs that integrate different functionalities such as OCR and TTS.

Development View
[This view focuses on the software development process, including the tools, technologies, and methodologies used.]

It includes the use of ReactJS and Tailwind CSS for frontend development, Python Flask for backend development, MySQL for database management, and open-source APIs for OCR and TTS functionalities.

Scenarios

Process View
[This view focuses on the dynamic aspects of the system, including how data flows through different processes and how different modules communicate.]

This includes data flow between components like OCR, TTS, and document conversion processes. It also captures user interactions and the flow of requests from the frontend to the backend.

Physical View
[This view deals with the system's deployment architecture and how software components are mapped to hardware resources.]

The physical view outlines how different components are deployed across servers, including the deployment of frontend and backend services, databases (MySQL), and microservices.
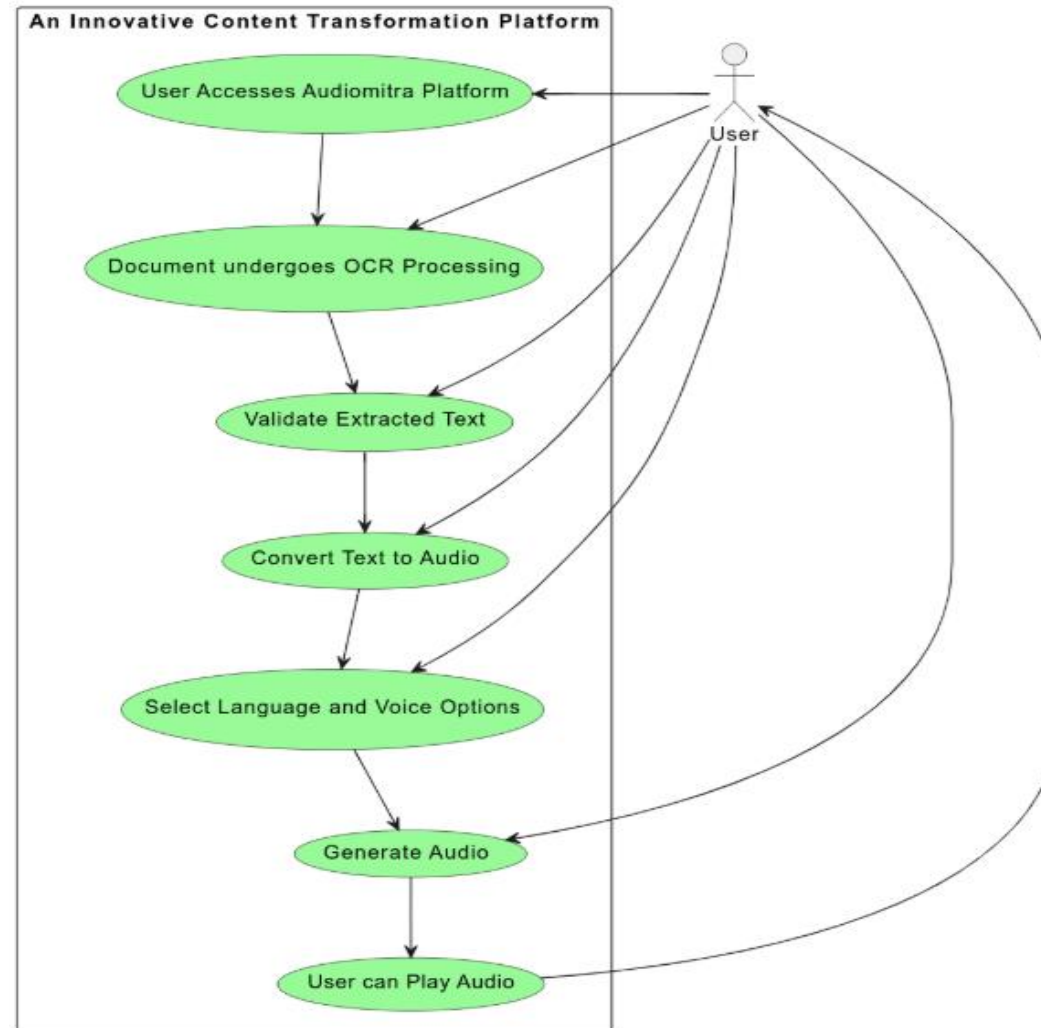
Use Cases

Examples of Use Cases:

- Converting printed documents into audiobooks.
- Digitizing a library's archive for preservation and accessibility.
- Providing multilingual support for educational content, enhancing accessibility and inclusivity.
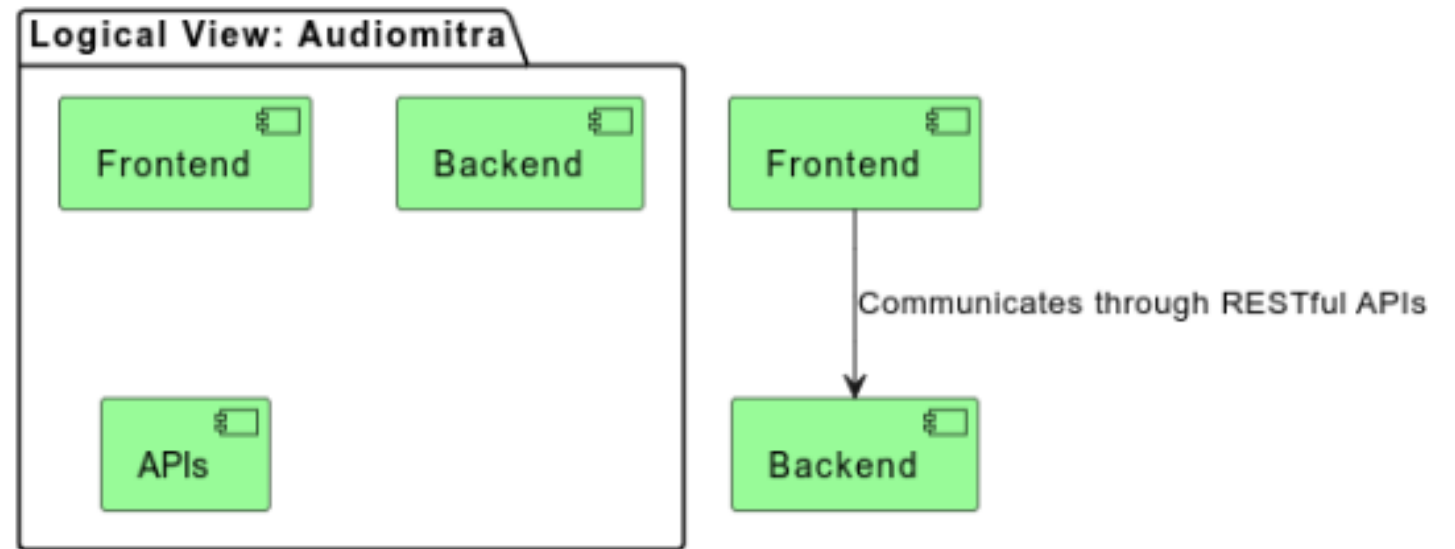
User Interactions

- Users access Audiomitra via the web interface.
- They initiate the conversion process by uploading scanned images or PDFs of printed documents.
- Document Processing:
    - Optical Character Recognition (OCR) extracts text from scanned images.
    - Extracted text undergoes validation for accuracy, allowing user review and edits if necessary.
- Text-to-Audio Conversion:
    - Validated text is converted into audio using Text-to-Speech (TTS) technology.
    - Users can select preferred languages and voice options for the audiobook.
- Audio Output:
    - The converted audiobook is available for download or streaming.
    - Users can listen to the audiobook on their preferred devices, enhancing accessibility and convenience.
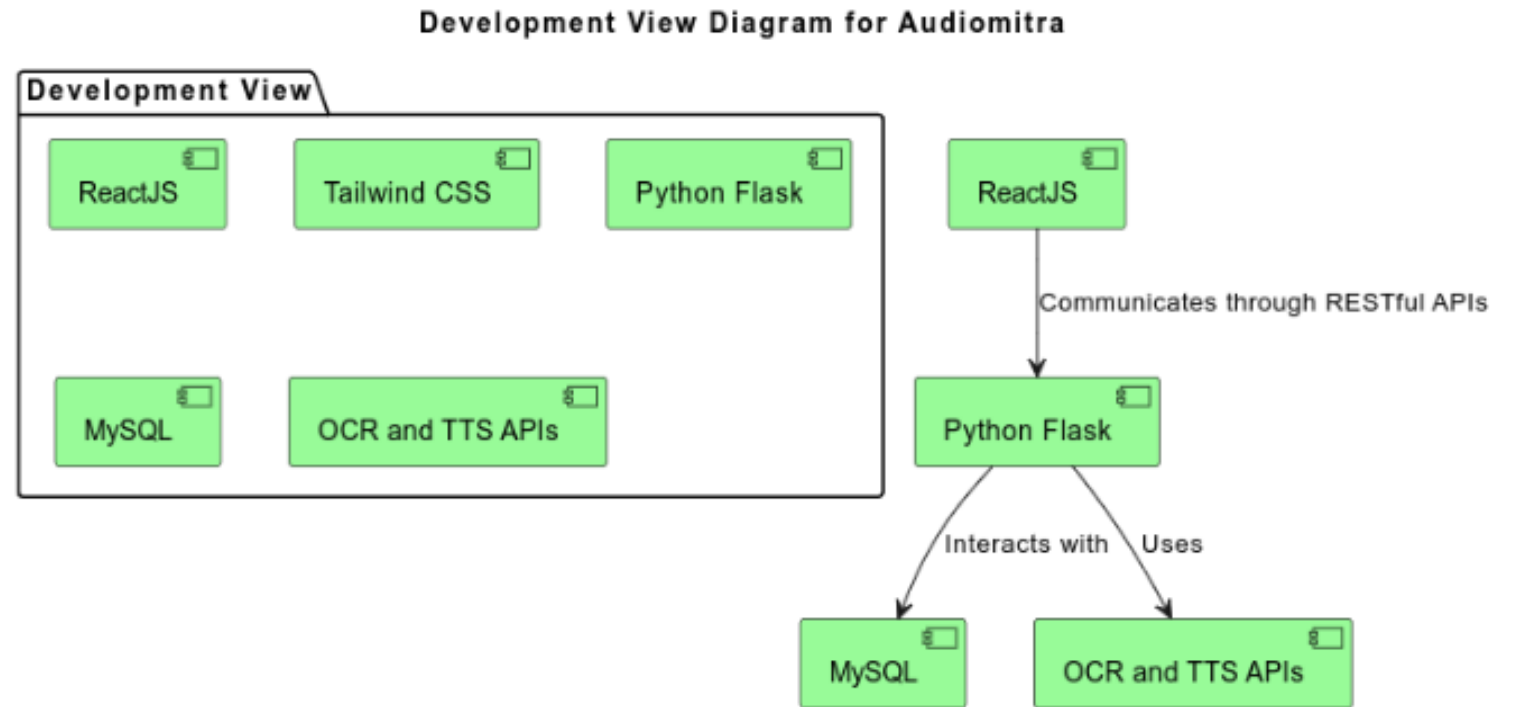
Use case diagram
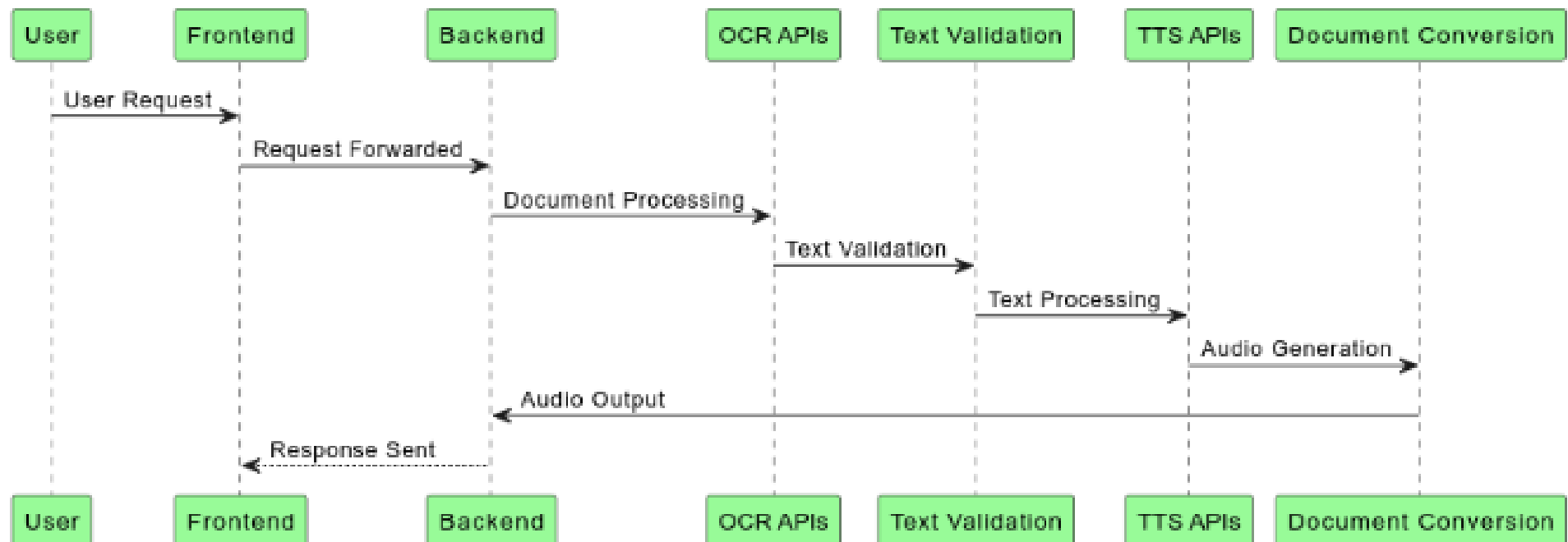
Logical view

Logical View Diagram for Audiomitra

Logical View: Audiomitra

Frontend

Backend

APIs

Frontend

Communicates through RESTful APIs

Backend

Audiomitra Logical View Diagram © 2024 by Vilal Ali & Team_11

Development View

Development View Diagram for Audiomitra

Development View

ReactJS	Tailwind CSS	Python Flask

MySQL	OCR and TTS APIs

ReactJS

Communicates through RESTful APIs

Python Flask

Interacts with   Uses

MySQL   OCR and TTS APIs

Audiomitra Logical View Diagram © 2024 by Vilal Ali & Team_11
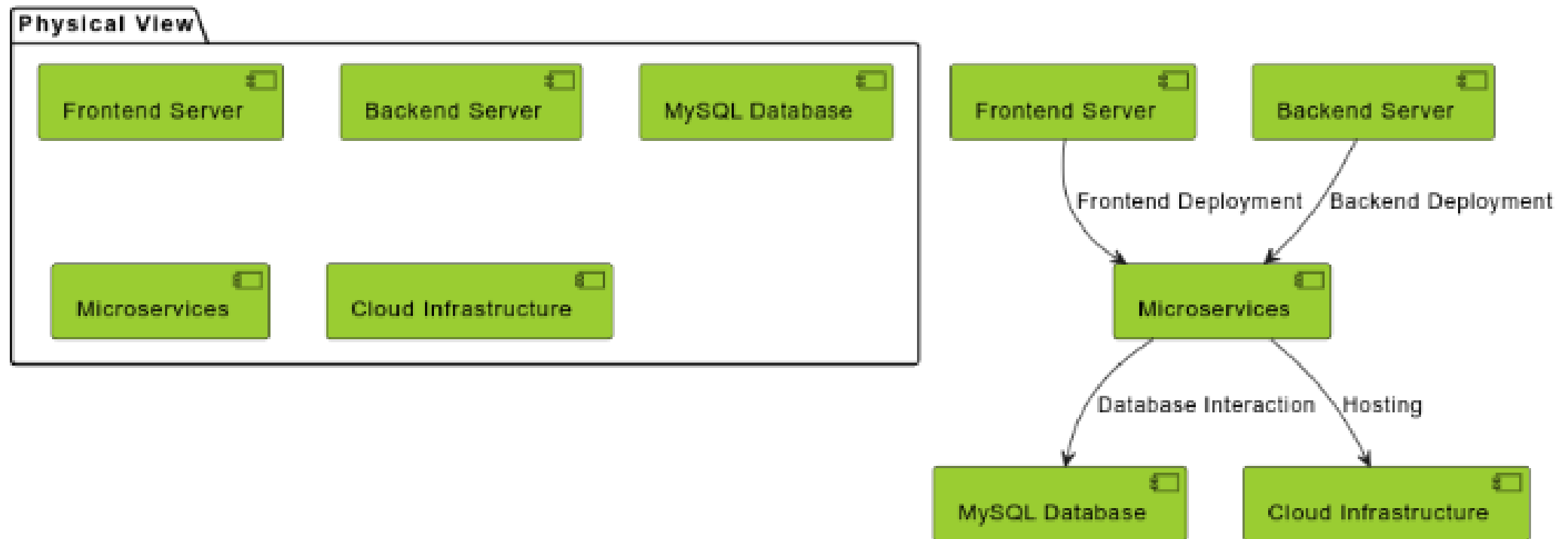
Process View

Process View Diagram for Audiomitra

Physical view

Physical View Diagram for Audiomitra

Subsystems

Subsystem Components for Audiomitra

Frontend Subsystem (Frontend)	OCR Processing Subsystem (Image to Text Processing)	Audio Output Subsystem (Audio Output)
UI/UX Subsystem (UI/UX)	Text Validation Subsystem (Text Validation)	Integration Subsystem (Integration)
Backend Subsystem (Backend)	TTS Processing Subsystem (Text to Audio Processing)	Database Subsystem (Database)
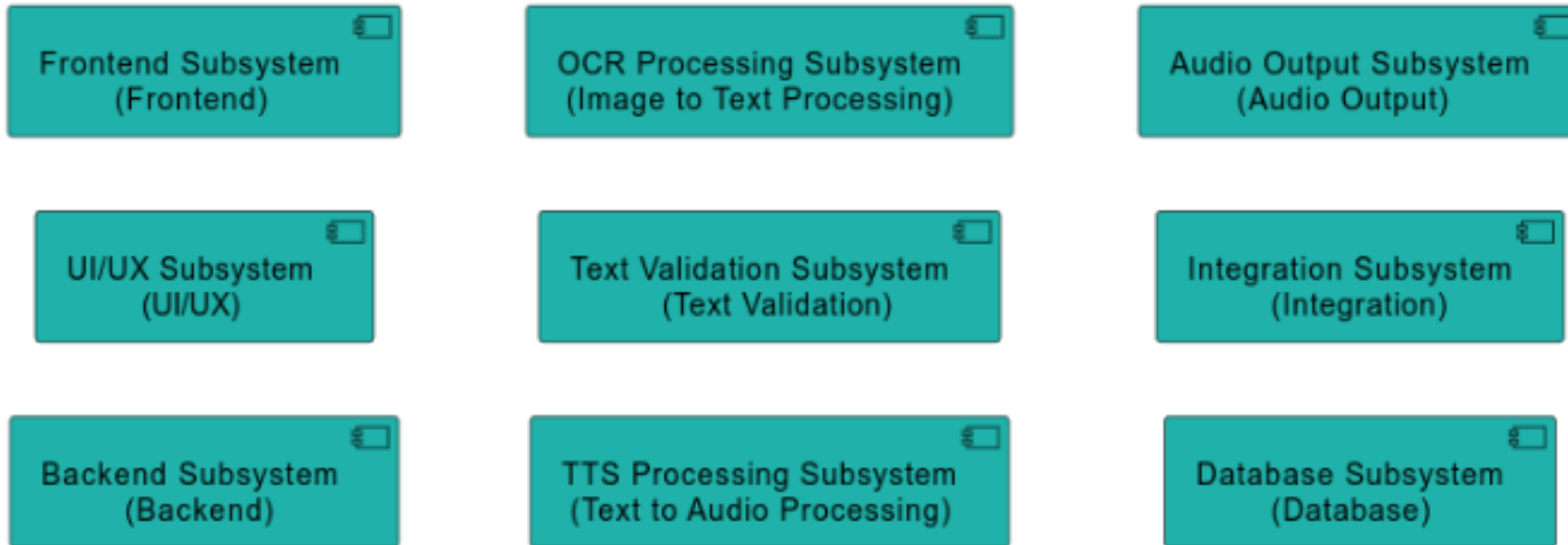
Architectural Tactics and Patterns

| Load Balancing	Caching	Data Replication	Failover	Service Decoupling
• Distributes workload evenly across servers, enhancing responsiveness and reliability.	• Temporarily stores frequently accessed data, reducing latency and improving data retrieval speed.	• Copies data across servers for enhanced availability and disaster recovery.	• Automatically switches to standby components upon active component failure, ensuring system reliability.	• Minimizes dependencies between components through interfaces, enhancing modularity and scalability.

Architectural Tactics

Benefits

- Enhances system responsiveness, reliability, and scalability.
- Improves performance through caching and load balancing.
- Ensures data availability, disaster recovery, and fault tolerance.
- Facilitates modular and scalable system architecture.

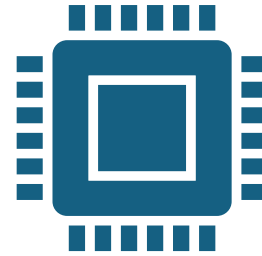# Microservices Architecture (MA) Pattern
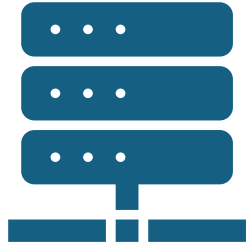
Description:

Segmentation of services (OCR, TTS, validation) into separate microservices.

API Gateway for routing client requests to the appropriate microservice.

Data storage options: Each microservice can have its database or share a common one based on data consistency needs.

Quality Attributes:

Scalability: Independent scaling of services based on demand.

Flexibility: Independent development, deployment, and updates of services.

Fault Isolation: Failure in one service doesn't impact others.

Maintainability: Ease of maintenance, modification, and enhancement.

Reliability: Consistent performance without errors or failures.
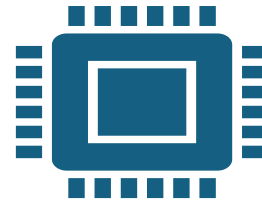
Event-Driven Architecture (EDA) Pattern

Description:

Event production and consumption for tasks like OCR, TTS, and validation.

Real-time processing for timely responses to user actions.

Decoupling of components for independent development, deployment, and scaling.

Quality Attributes:

Real-time Processing: Timely responses to user actions and updates.

Decoupling: Independent development, deployment, and scaling of components.

Scalability: Horizontal scaling to handle increasing volumes of events and data.

Reliability: Minimize the risk of data loss or processing errors with fault-tolerance mechanisms.

Fault Tolerance: Graceful handling of errors and failures, ensuring system operability.

A comparative study

Microservices Architecture (MA)

- Pros:
  - Scalability: Independent scaling for efficient resource allocation.
  - Flexibility: Independent development, deployment, and updates.
  - Fault Isolation: Failures in one service don't impact others.
  - Maintainability: Modular design for easier maintenance and enhancement.
  - Reliability: Isolation of failures improves system reliability.
- Cons:
  - Complexity: Managing multiple services introduces complexity.
  - Latency: Inter-service communication may introduce latency.
  - Operational Overhead: Requires robust infrastructure and monitoring.

Event-Driven Architecture (EDA)

- Pros:
    - Scalability: Asynchronous communication for efficient scaling.
    - Flexibility: Loose coupling allows independent evolution.
    - Fault Isolation: Decoupled components reduce failure impact.
    - Maintainability: Components can be updated without system disruption.
    - Real-time Processing: Suitable for systems with sporadic events.
- Cons:
    - Complexity: Event schema design and management can be complex.
    - Eventual Consistency: Ensuring consistency across components can be challenging.
    - Monitoring and Debugging: Debugging asynchronous systems can be more challenging.

Summary

Microservices Architecture (MA) offers scalability, fault isolation, but with complexity.

Event-Driven Architecture (EDA) provides scalability, real-time processing, but with schema and monitoring challenges.

Choose based on system needs: MA for clear boundaries, EDA for real-time processing and flexibility.

Contribution Table

S. N.	Task Name	Implemented Design Pattern	Asignee Name	Contribution
1	Task 1: Requirements and Subsystems	Functional and Non-functional Requirements.	Vilal	Vilal 100%
		Subsystem Overview		
2	Task 2: Architecture Framework	Stakeholder Identification	Madan	Madan 60% Vilal 20% Team 20%
		Major Design Decisions		
3	Task 3: Architectural Tactics and Patterns	5 Architectural Tactics employ in Audiomitra	Shriom	Hanuma 50% Shriom 30% Vilal 10% Team 10%
		Implementation Patterns	Hanuma	
4	Task 4: Prototype Implementation and Analysis	Prototype Development	Team	Vilal 40% Hanuma 20% Madan 20% Shriom 20%
		Architecture Analysis		
5	Project Report Writing	##	Vilal	Vilal 40% Shriom 20% Hanuma 20% Madan 20%
6	Project Repo Link	https://github.com/vilalali/audioMitra		
7	Other PR Link	https://github.com/vilalali/audioMitra/tree/task4_architectural_patterns		

Conclusion

- Both Microservices Architecture (MA) and Event-Driven Architecture (EDA) have distinct advantages and challenges.

- MA offers control over scalability, fault isolation, and reliability, suitable for diverse components like document processing.

- EDA provides flexibility, real-time processing, and fault isolation, ideal for sporadic events.

- Choose MA for clear control and reliability, and EDA for flexibility and real-time processing.

- The "Document Processing Subsystem" will adopt Microservices Architecture (MA) for scalability and reliability, aiming for a high-performance system.

It's not a bug; it's an undocumented feature.
- Anonymous

Thank you!