ACKNOWLEDGEMENT

I am greatly indebted to my internship guide **Dr. A. M. Rajurkar** for her able guidance throughout this work. It has been an altogether different experience to work with her and I would like to thank her for help, suggestions and numerous discussions.
I gladly take this opportunity to thank **Dr. A. M. Rajurkar** (Head of Computer Science & Engineering, MGM's College of Engineering, Nanded).
I am heartily thankful to **Dr. Lathkar G. S.** (Director, MGM's College of Engineering, and Nanded) for providing facility during progress of project, also for her kindly help, guidance and inspiration.
Last but not least I am also thankful to all those who help directly or indirectly to develop this seminar and complete it successfully. With Deep Reverence,

Piyush Mahajan (223)

ABSTRACT

This internship report presents a comprehensive account of my full-time professional experience at Tangible IT, Mexico, where I served as an AI-Driven Full Stack Developer. The internship provided hands-on exposure to enterprise-grade application development through two major platforms—AI Videos and AI Discover—both of which leverage artificial intelligence to enhance content processing, educational support, and intelligent conversational interactions.

My role involved working on the complete development lifecycle of both platforms, including frontend UI development, backend logic integration, API orchestration, database configuration with CosmosDB, and real-time cloud deployment strategies. AI Videos focused on transcript extraction, document analysis, and knowledge enhancement features like summarization, flashcards, mind map generation, and translation. AI Discover, on the other hand, was a dynamic conversational system integrated with web search, image retrieval, and persistent history handling—designed to support natural multi-turn dialogues with contextual awareness.

Throughout this internship, I contributed to improving platform performance, implementing advanced features such as input-type transitions, dark mode, and error-handling mechanisms, while adhering to agile workflows and version control practices. My journey at Tangible IT not only strengthened my technical proficiency in full-stack development but also equipped me with crucial industry skills like problem-solving, cross-functional collaboration, and scalable solution design. This experience has been instrumental in shaping my professional capabilities and laying a strong foundation for future contributions to the field of intelligent application development.

TABLE OF CONTENTS

List of Figures

Chapter 1

INTRODUCTION

This chapter provides a foundational overview of the full-time internship experience undertaken at Tangible IT, Mexico, where I was appointed as an AI-Driven Full Stack Developer. The internship not only marked a significant step in my professional journey but also offered immersive exposure to enterprise-grade application development, AI integration, and digital transformation practices. Over the course of the internship, I worked closely with senior developers and product strategists to contribute to two key products—AI Videos, a content analysis and summarization platform, and AI Discover, a conversational AI system with advanced knowledge retrieval features.

The following sections elaborate on the organization's background, the internship objectives, and the scope of work undertaken during the internship period. Through this experience, I acquired critical insights into AI-driven system development, production deployments, optimization techniques, UI/UX refinement, and scalable architecture design—all of which have shaped my technical capabilities and professional mindset.

1.1 Company Background

The rapid advancement of technology has transformed the landscape of the IT industry, making internships a critical bridge between academic knowledge and real-world application. Internships today do not just expose students to the corporate work culture but also enable them to engage with live industrial problems, propose solutions, and contribute meaningfully to impactful projects. As a final-year engineering student specializing in Computer Science, I had the privilege of completing a full-time internship with Tangible IT, Mexico—a globally recognized consulting and systems integration firm that focuses on digital transformation through cutting-edge technologies and strategic IT services. My role at Tangible IT was that of a Full-Time AI-Driven Full Stack Developer under the guidance of Luis Angel Silva, the company's Director. I worked six days a week on live, client-level projects in both frontend and backend environments, ensuring complete immersion in the development lifecycle.

Tangible IT, headquartered in Mexico, is a globally recognized consulting and systems integration company that specializes in driving digital transformation for modern enterprises. With a strong foundation built on expertise, client trust, and innovation, Tangible IT enables organizations to adapt to rapidly evolving technological landscapes through tailored solutions.

Fig No. 1.1: Company Logo

The organization prides itself on its holistic approach to solving real-world business problems. Rather than delivering isolated services, Tangible IT embeds itself within the core operational strategies of client companies to offer end-to-end digital solutions—from system consultation and cloud integration to data analytics, enterprise applications, and mobility platforms.

Tangible IT's commitment to quality and excellence is reflected through its strong partnerships with some of the world's most reputable technology providers. These alliances strengthen its delivery capabilities across various sectors and technical domains.

Key Domains of Expertise:

Tangible IT delivers scalable, secure, and future-proof services in areas such as:

- **Cloud Solutions & Integration**: Migrating legacy infrastructures to modern, scalable cloud environments (Azure, AWS, etc.).
- **Data Analytics**: Implementing real-time and batch-processing analytics pipelines that help clients gain actionable insights.
- **Enterprise Resource Planning (ERP)**: Deploying customized ERP solutions to streamline complex organizational workflows.
- **Digital Services & Transformation**: Revamping end-user experience through UI/UX design, mobile-first platforms, and intelligent systems.
- **Security & Compliance**: Ensuring robust cybersecurity implementations and adherence to global standards.

Partnerships That Empower Innovation:

Tangible IT maintains strategic collaborations with:

- **Microsoft**: For cloud infrastructure, Azure integrations, and Office 365-based enterprise tools.
- **Amazon Web Services (AWS)**: To deploy scalable, cost-effective backend environments.

- **SAP**: For robust enterprise-grade ERP solutions.
- **Salesforce**: For advanced CRM solutions and business automation platforms.

Vision, Mission, and Values:

- **Mission**: To deliver *tangible value* through strategic IT transformation that accelerates operational excellence and business innovation.
- **Vision**: To be the most reliable transformation partner that understands and aligns with every client's long-term ambitions.
- **Core Values**: Integrity, innovation, collaboration, and continuous learning. Tangible IT believes in co-creating solutions with clients, ensuring mutual growth and long-term success.

Organizational Culture:

The work culture at Tangible IT is fast-paced yet collaborative, encouraging engineers and developers to think creatively while remaining grounded in industry best practices. Employees are empowered to take ownership of their work, contribute ideas, and engage in continuous upskilling. Daily stand-ups, review sessions, agile methodologies, and transparent communication help build a growth-driven, professional environment.

By combining technical depth with business foresight, Tangible IT is not just a service provider but a strategic technology partner for its clients.

It continues to play a key role in driving intelligent digital ecosystems, enabling companies to remain competitive, efficient, and future-ready.

1.2 Project Overview

During my tenure at Tangible IT, Mexico, I was assigned to work on two high-impact, AI-powered applications — AI Videos and AI Discover. These projects were not isolated software pieces, but integral components of a larger strategy to develop intelligent digital platforms that assist learners, researchers, and enterprises in deriving structured insights from unstructured data sources.

AI Videos

The AI Videos platform is designed as an intelligent content summarization and knowledge transformation tool. It allows users to extract transcripts from YouTube videos, analyze uploaded PDFs, and interpret text from images, transforming these diverse data formats into AI-generated summaries, flashcards, mind maps, and translated outputs. Its purpose is to reduce the cognitive load on learners by converting long-form content into concise, structured, and interactive formats.

The system is feature-rich and supports:

- Transcript Extraction from video content using public URLs.
- PDF Document Analysis using deep parsing and summarization techniques.

- Image Content Analysis by applying OCR and vision models to extract and describe image data.
- Language Translation, Flashcard Generation, Mind Mapping, and Text-to-Speech features for accessible learning.

My contributions here spanned UI implementation, backend integration, and AI workflow management. I worked on creating and managing tab transitions between input types (YouTube, PDF, Image), ensuring smooth output rendering, optimizing state resets, and improving UI responsiveness for both mobile and desktop. I also handled challenges related to history persistence, multi-format data handling, and ensuring that user sessions reflected accurate and clean results after each operation. I implemented logic that reset the view to the summary tab upon every new request, removed residual outputs, and ensured all transitions were seamless.

Additionally, I proposed enhancing the platform's branding, suggesting a name more aligned with its diversified capabilities beyond just video summarization. I also contributed significantly to internal documentation to assist future developers in understanding the platform's architecture and workflows.

AI Discover

The AI Discover project functions as a smart assistant — a chatbot platform enhanced with real-time web search, image search, document memory, and CosmosDB-backed history persistence. Unlike traditional chatbots that operate on static data, AI Discover dynamically fetches updated information from the web and returns well-structured answers, complete with citations, tool-based responses, and interactive media when applicable.

Key features include:

- Natural language input support with intelligent tool selection (web search, image generation, etc.)
- Smart image query detection and inline image rendering with shadow effects and rounded cards
- CosmosDB-based persistent conversation history with intelligent title generation
- Deep integration with ACV app context using conditional parameterization

(isFromACV)

- Support for anonymous and authenticated users with differing permissions (e.g., mass deletion)

I contributed to the overall system flow by working on:

- Backend handling of tool responses (user > tool > assistant message structure)
- History logic involving unique slugs, email-based tracking, and session-level memory
- Dealing with schema errors and performance delays during large query processing
- Implementing a robust conversation recovery system where users could resume prior chats with accurate context
- Optimizing CosmosDB querying and storage, ensuring chat threads and titles were updated without redundancy

This project enhanced my backend engineering skills, particularly in integrating multiple data models, streamlining AI response pipelines, and ensuring scalability and fault tolerance across environments.

Together, both AI Videos and AI Discover represent enterprise-grade applications developed with a focus on usability, performance, and intelligent AI-human interaction. These platforms not only showcase the convergence of frontend and backend development but also highlight the power of AI in transforming content consumption and information retrieval. My involvement in these applications gave me valuable exposure to real-world software development at scale, from ideation to deployment, and equipped me with advanced skills in full-stack AI-driven systems.

1.3 Objectives

The primary objective of undertaking this full-time internship at **Tangible IT, Mexico** was to gain in-depth exposure to enterprise-level application development with a strong focus on artificial intelligence integration, full-stack implementation, and real-time problem-solving. As a final-year engineering student specializing in Computer Science and Engineering, I aimed to bridge the gap between academic understanding and industrial practice through immersive participation in high-impact digital transformation projects.

This internship offered a rare opportunity to work directly under the guidance of experienced professionals such as **Luis Angel Silva**, Director at Tangible IT, and contribute to **live production-grade systems** that solve modern-day content processing and knowledge dissemination challenges.

Academic Objectives

Apply theoretical knowledge in practical environments:

I aimed to implement core concepts learned during my coursework — such as database systems, web technologies, software engineering principles, and AI fundamentals — in solving real-world business problems.

Develop real-time software components:

Unlike classroom assignments or capstone projects, I wanted to contribute to actual systems with real users, deadlines, and performance expectations. This objective included writing scalable, efficient code and collaborating with cross-functional teams.

Learn and apply the software development life cycle (SDLC):

A crucial goal was to understand the end-to-end lifecycle of enterprise applications — from ideation, requirement gathering, prototyping, development, testing, deployment, and maintenance — with particular emphasis on Agile development methodologies.

Understand AI-driven platforms in production settings:

The internship allowed me to not only experiment with AI models but understand how they are integrated into applications at scale. I explored text summarization, speech-to-text, translation, OCR, and chatbot architectures.

Technical Objectives

Master Full-Stack Development:

My role as an AI-Driven Full Stack Developer demanded proficiency in both frontend and backend. I set a goal to deepen my understanding of React.js, Node.js, CosmosDB, and REST APIs, along with UI responsiveness and state management.

Understand cloud-based architecture:

With Tangible IT being a Microsoft Azure partner, a key goal was to learn about cloud deployment, serverless architecture, and CosmosDB configurations for distributed data storage and management.

Work with production-level databases and real-time data pipelines:

I specifically aimed to learn how to configure document schemas, optimize database queries, and resolve integration challenges with tools like CosmosDB and cloud APIs.

Gain hands-on experience in debugging and optimization:

Whether it was backend logic failure, API rate limits, schema mismatch, or UI transition bugs, resolving real-time production issues helped me refine my skills in critical thinking, root cause analysis, and code optimization.

Professional Development Objectives

Adapt to industrial coding standards and collaboration tools:

My aim was to follow proper version control using Git, maintain documentation, conduct peer code reviews, and follow structured coding and naming conventions used in professional environments.

Improve communication and collaboration in a remote team:

Since the internship was guided remotely, it became essential to master remote collaboration, team discussions, and documentation clarity. I learned how to raise technical issues, incorporate feedback, and coordinate effectively with team members across time zones.

Develop product thinking and end-user empathy:

Working on platforms like AI Videos and AI Discover helped me consider not just functionality

but user experience, accessibility, branding, and interface adaptability. I gained insights into how enterprise tools must solve real problems in the simplest, most intuitive way possible.

Build long-term readiness for the tech industry:

Lastly, I aimed to use this experience as a stepping stone toward building a strong resume, sharpening my portfolio, and gaining the confidence to contribute in any tech-focused organization with readiness and professionalism.

This internship wasn’t just a requirement for academic credits — it was a strategic step toward professional excellence. Each line of code I wrote, every deployment I contributed to, and every issue I resolved aligned with my broader objective of becoming a competent, creative, and industry-ready software developer.

1.4 Scope of Work

The scope of my internship at Tangible IT, Mexico as an AI-Driven Full Stack Developer extended across a broad spectrum of software engineering practices. I was engaged in real-time product development cycles, cloud-based application deployments, and AI-powered platform integrations, contributing actively to both frontend and backend layers of two major projects: AI Videos and AI Discover.

This internship was not limited to minor fixes or theoretical tasks. Rather, it involved comprehensive participation in high-impact development workflows, problem resolution in production environments, and performance optimization of live platforms designed for educational and professional ecosystems.

Full Stack Development

As a full stack developer, I worked across multiple layers of both AI platforms:

- Frontend Responsibilities:
    - Designed and implemented reusable UI components using React.js.
    - Managed dynamic layout rendering based on input types (YouTube URL, PDF, Image).
    - Integrated accessibility features such as dark/light mode and text-to-speech.
    - Worked on tab navigation, content rendering, and context-specific visual transitions.
    - Ensured responsiveness and cross-platform compatibility of the interfaces.
- Backend Responsibilities:
    - Integrated and extended backend APIs using Node.js.
    - Handled logic for AI-based summarization, translation, and flashcard generation.
    - Created dynamic routes for processing content types and managing history data.
    - Implemented state-clearing logic when switching between modules to avoid stale data propagation.
    - Participated in logic flow optimization to reduce response latency.

Database Design & Integration

- Managed CosmosDB, a globally distributed NoSQL database, for storing chat histories, user data, and session details.
- Contributed to schema design, slug-based history storage, and secure conversation referencing.
- Resolved issues related to document updates, malformed references, and inefficient indexing.
- Participated in critical discussions on data normalization, backup policies, and compliance handling for user-specific data storage.
- Ensured reliable data fetching using slugs and unique identifiers, and optimized query performance.

AI-Powered Feature Implementation

Both AI Videos and AI Discover featured AI components at their core. I was actively involved in:

- Integrating text summarization models for transcripts, PDFs, and image-extracted text.
- Building flashcard and mind map generators from extracted content using NLP pipelines.
- Handling real-time translation and multi-language UI support.
- Improving web search-based Q&A modules by implementing structured answer rendering.
- Working with visual content descriptors and image search components using intelligent filters.

Deployment & Pre-production Readiness

- Participated in multiple pre-production and production deployments.
- Managed incremental backend deployments for minimizing downtime and ensuring rollback safety.
- Monitored logs, handled version conflicts, and collaborated on error triaging during release cycles.
- Contributed to deployment documentation and team-wide communication regarding updated workflows.
- Handled platform state persistence to ensure user continuity across sessions.

Feature Proposals & Branding Inputs

- Proposed enhancements like renaming “AI Videos” to better reflect feature expansion.
- Suggested UX changes like auto-reset to summary tab on new input uploads.
- Created structured documentation and demo scripts to support internal review and client demonstration.
- Provided written documentation for newly added modules and internal logic references.

Team Collaboration & Workflow Participation

- Regularly participated in team meetings and sprint planning with the Director and senior developers.
- Incorporated peer feedback into development cycles.
- Used Git for version control, collaborated via GitHub repositories, and resolved pull request issues.
- Reported bugs and improvement areas after every deployment cycle and proactively shared solutions.

- Gained experience in cross-functional collaboration while communicating with AI model trainers, UI/UX designers, and deployment engineers.

The scope of this internship was highly extensive, providing exposure to diverse areas of modern enterprise software development. It was not only a technical engagement but a strategic experience that sharpened my understanding of collaborative innovation, production scalability, and AI-integrated product design.

Chapter 2

AI VIDEOS PROJECT

The AI Videos platform represents one of the flagship projects developed during my internship at Tangible IT, Mexico, as part of my role as a Full-Time AI-Driven Full Stack Developer. This project aims to redefine how users interact with long-form digital content—whether in the form of YouTube videos, PDFs, or image files—by converting it into simplified, structured, and learning-ready formats using Artificial Intelligence. Designed as a content intelligence and summarization tool, AI Videos addresses the growing need for faster knowledge consumption in the era of information overload.

Built with a seamless user interface and a robust backend architecture, this platform utilizes advanced technologies such as NLP-based summarization, image recognition, text extraction, mind map generation, and flashcard synthesis to help learners, educators, and researchers easily comprehend multimedia material. The AI Videos project encapsulates the spirit of digital transformation by not just providing access to content but helping users understand and retain it more efficiently.

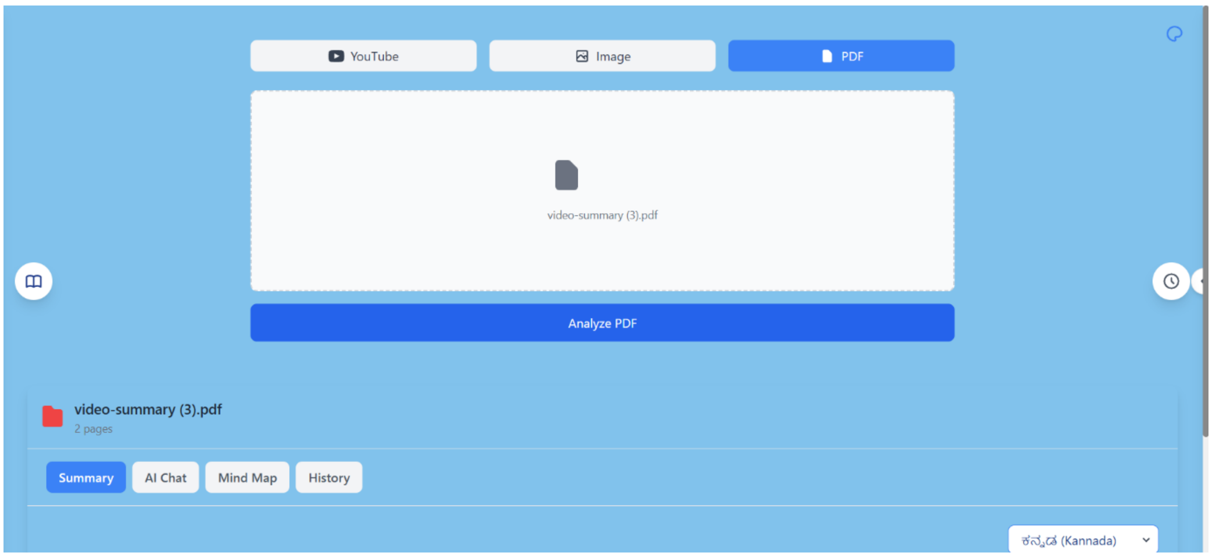


Fig No. 2.1: AI Videos Project Interface

2.1 Objectives of AI Videos Platform

The core objective behind developing AI Videos was to empower users with an intelligent platform that simplifies unstructured multimedia content into digestible educational units using Artificial Intelligence.

The platform targets a broad audience—ranging from students and teachers to working professionals—who seek to save time and improve learning outcomes by accessing concise, context-rich information from vast data sources like YouTube lectures, academic PDFs, and text-heavy visuals.

Some of the key goals of AI Videos include:

Automate and Simplify Content Extraction from Multimedia Sources

The platform was designed to automatically extract transcripts from YouTube videos, scan PDF documents, and recognize text from images—thus removing the need for manual transcription or note-making. Whether it's an educational lecture, research paper, or infographic, users can submit the content and receive a structured transcript within seconds.

This automation saves hours of manual work for educators, students, and analysts, ensuring that content is always accessible, searchable, and ready for processing.

The intelligent parsing algorithms maintain the structure of source documents and videos by preserving timestamps, page numbers, and visual sections.

Enhance Learning through AI-Powered Summarization

A major objective of AI Videos was to go beyond transcription and provide AI-generated summaries of long-form content. These summaries are not just surface-level; they are crafted using Natural Language Processing (NLP) to extract the most relevant insights, arguments, and facts from the input material.

The platform aims to replicate a "smart study companion" that can tell the user what matters most from a 45-minute video or a 30-page document, within a fraction of the time. This is especially beneficial for students preparing for exams or professionals trying to stay updated with industry knowledge.

Generate Interactive Study Aids for Knowledge Retention

One of the unique selling points of AI Videos is its ability to convert raw content into interactive learning formats. The system automatically generates

- Flashcards with questions and answers derived from the source content
- Mind Maps that visually represent key concepts and their relationships
- Bullet-point lists for quick revision

These tools turn the passive act of watching or reading into an active learning process. They help users engage with the material, test their understanding, and retain knowledge more effectively.

Support Multi-format Input Types and Seamless Switching

Unlike single-purpose applications, AI Videos was built with the flexibility to handle multiple input types—including video URLs, PDF uploads, and image files. A smart input-detection system ensures that users can switch between input modes without disrupting previous outputs.

The system clears previous sessions and resets the UI state each time a new input type is selected, offering a clean and intuitive user experience. This modular design makes the application versatile and suitable for a range of user needs and content types.

Improve Accessibility with Multi-language and Text-to-Speech Features

AI Videos incorporates multi-language processing, enabling transcript generation, summarization, and translation across various global and regional languages. The platform also includes Text-to-Speech (TTS) capabilities that allow users to listen to summaries and transcripts—especially helpful for visually impaired users or multitaskers.

With these features, AI Videos aligns with global accessibility standards and ensures that language is never a barrier to knowledge. The interface supports localized language settings, and audio playback controls are provided for a user-friendly experience.

Deliver a Smooth, Responsive, and Interactive User Experience

A critical objective was to offer a responsive UI that is both visually appealing and functionally efficient. Features like tabbed navigation, collapsible sidebars, loading indicators, and dark/light mode were added to enhance usability.

The interface adapts to various screen sizes and devices, allowing users to interact with the platform on desktops, tablets, or smartphones. This ensures accessibility across different user environments—from students in rural areas with mobile access to researchers using high-end desktop systems.

Enable Persistent Storage and Export Options

To enhance workflow continuity, AI Videos enables persistent session management using cloud-based storage. Users can revisit previous sessions, download their content in PDF or PPT format, and export study aids for offline access.

This ensures that users have a digital archive of their learning materials and insights. It also makes the platform suitable for long-term use in academic institutions, LMS integrations, or enterprise knowledge portals.

Align Product Design with Real-World Use Cases and Educational Needs

The project was conceived to fulfill actual needs faced by students, educators, and working professionals who engage with large volumes of multimedia content daily. Through every stage of the design and development process, the team ensured that each feature solves a real problem and adds tangible value.

AI Videos is not just a tech demo—it is a full-fledged product prototype designed to meet learning, accessibility, and knowledge extraction goals at scale.

In essence, AI Videos was not just an AI project but a user-centric product that sought to blend automation, accessibility, and personalized learning into one unified application. These goals guided every phase of its development—from ideation and architecture to feature enhancement and final deployment.

2.2 System Architecture and Workflow

The system architecture of AI Videos is a sophisticated multi-layered design that integrates frontend, backend, AI models, and cloud infrastructure into a unified, scalable platform. The architecture was carefully constructed to ensure real-time performance, modular extensibility, high fault tolerance, and user-friendly interaction. The primary workflow follows a structured pipeline from content input to output generation and user interaction.

Frontend Layer (User Interaction & Input Management)

The frontend acts as the interactive bridge between the user and the AI processing layers. It is built using modern frontend frameworks such as React.js, designed to offer responsiveness, accessibility, and clarity. The interface presents users with clear input options including YouTube video links, PDF uploads, and image file inputs—each routed through a separate tabbed interface.

Features of the frontend include:

- Dynamic Input Routing: Allows switching between content types without reloading the page.
- Dark/Light Mode Toggle: Enhances usability across various lighting environments.
- Session-based History View: Displays the user’s past interactions, summaries, and generated flashcards.
- Live Feedback Indicators: Shows processing status, loading animatioconfirmation

Backend Layer (Data Processing & AI Logic)

The backend, developed primarily using Node.js and Express, acts as the control center for handling business logic, AI calls, session management, and API orchestration. It is integrated with:

- Cloud-based AI Models: To perform NLP-based summarization, entity recognition, translation, and flashcard generation.
- File Handlers: For extracting structured text from PDFs and recognizing text from image files using OCR (Optical Character Recognition).
- Service Workers: To handle asynchronous data fetching, prevent UI blocking, and improve efficiency through progressive loading.

The backend pipeline includes:

- Input Preprocessing: Verifies content type and sanitizes input (e.g., validating YouTube links or file types).
- Task Dispatcher: Routes the data to the correct processing engine—Whisper for audio transcription, GPT for summarization, Tesseract for OCR.
- Post-Processing: Structures the output into modular sections (summary, flashcards, timestamps).
- Error Handling: Captures failed states like "no transcript found" or "invalid PDF file" and sends fallback suggestions to the frontend.

AI Integration Layer (Core Intelligence Engine)

This layer integrates powerful AI capabilities to derive meaning from content. The platform uses a combination of:

- OpenAI's GPT Models: For natural language understanding and summarization.
- Whisper: For speech-to-text processing of video audio.
- Custom-trained Transformers: For flashcard and mind map generation.
- Google Translate API / Azure Translator: For multilingual support.

Notable features of this layer include:

Multi-turn NLP Processing: To support follow-up questions on the summarized data.

- Topic Detection and Clustering: Grouping similar concepts to form flashcards and mind maps.

- Redundancy Filters: Removes repetitive or irrelevant transcript data before summarization.
- Contextual Summarization: Ensures output preserves temporal structure and flow from the original content.

Database & Session Management (Using CosmosDB)

A critical part of the architecture is the use of CosmosDB, a scalable NoSQL database that ensures fast read/write operations and seamless cross-platform session syncing. The database is used to:

- Store transcript data, summaries, and flashcards mapped to each session.
- Preserve user history across devices using email and slug identifiers.
- Enable secure and structured deletion of records (per message or full history).
- Generate intelligent titles based on initial summary content or user input.

Improvements made during development include:

- Slug-based Indexing: For faster conversation reloads via /chat/:id.
- History Pagination: For performance-optimized history views.
- Multi-format Record Mapping: Each session stores video, PDF, and image data independently to avoid mixing outputs.

Deployment & Cloud Hosting

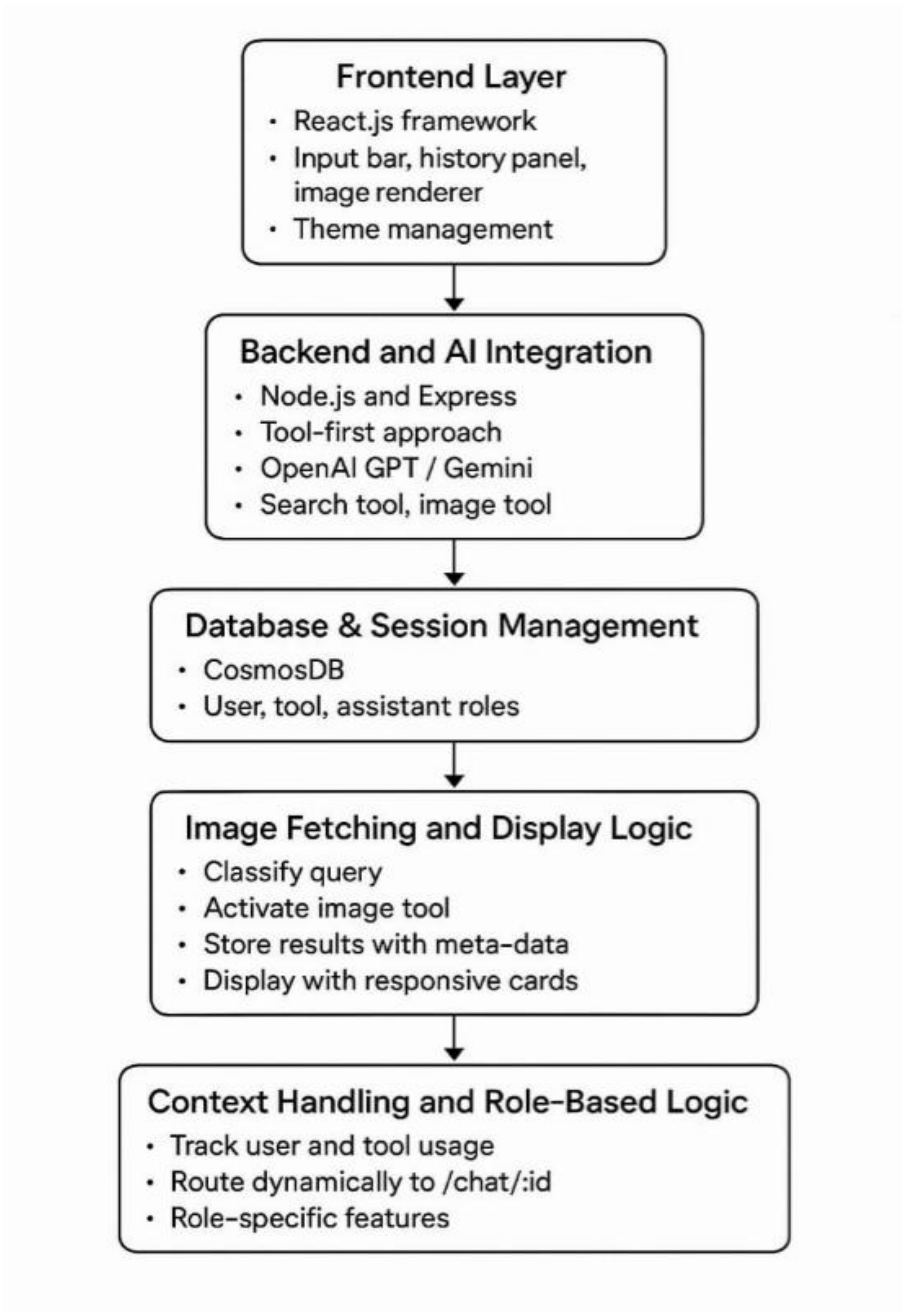
The platform is deployed using Azure Cloud Infrastructure, ensuring global availability, fault tolerance, and fast scalability.

The architecture supports:

- CI/CD Pipelines: Automated deployment to staging and production environments.
- Load Balancers: For distributing user requests during peak times.
- Environment-Based Logging: To handle separate error tracking in pre-production and production.

Additional components include:

- Content Export Services: For exporting outputs as PDF or PPT using serverless functions.
- Performance Monitoring: Using Application Insights or equivalent tools for log tracing and performance metrics.



Frontend Layer
• React.js framework
• Input bar, history panel, image renderer
• Theme management
Backend and AI Integration
• Node.js and Express
• Tool-first approach
• OpenAI GPT / Gemini
• Search tool, image tool
Database & Session Management
• CosmosDB
• User, tool, assistant roles
Image Fetching and Display Logic
• Classify query
• Activate image tool
• Store results with meta-data
• Display with responsive cards
Context Handling and Role-Based Logic
• Track user and tool usage
• Route dynamically to /chat/:id
• Role-specific features

Fig No. 2.2: System Architecture Overview AI Videos Project

This architecture diagram outlines the complete flow of the AI-integrated application, starting from the frontend interface built with React.js, progressing through backend AI tools, database management with CosmosDB, and ending with role-based logic handling. It showcases how each layer contributes to seamless interaction, real-time image processing, and personalized user experience.

Workflow Pipeline (Step-by-Step)

The AI Videos platform operates via a well-defined workflow that activates different modules based on user interaction:

User Input

The user selects a content type tab (YouTube, PDF, or Image), provides a valid input (e.g., video URL or document), and submits the request via the UI.

Input Validation & Dispatch

Frontend scripts validate input type and format. The data is then dispatched to the backend via an API call with contextual metadata (input type, session ID, language preferences).

Content Extraction

- If YouTube: Transcript is extracted using an external API (like youtube_transcript_api or yt-dlp + Whisper)
- If PDF: Text is parsed using pdfplumber or PyMuPDF.
- If Image: Text is extracted using Tesseract OCR.

AI-Powered Processing

Extracted text is sent to NLP engines. Based on user selections, different AI pipelines are triggered

- Summarization (Google Gemini or OpenAI models)
- Flashcard Generation (key-value QA pair synthesis)
- Mind Map Creation (concept hierarchy + visual structure)

Result Rendering

Processed content is returned to the frontend and displayed in dynamically generated tabs. Each tab supports scrolling, responsive UI, and theme adjustments.

Session Persistence

The entire interaction is saved in CosmosDB, tagged with session ID and user metadata for future access or export (PDF/PowerPoint).

Workflow Summary

1. User Uploads/Inputs Content
2. Frontend Sends API Request → Backend validates → Preprocessing
3. AI Model Triggered → Summarizes content, generates learning modules
4. Formatted Output Displayed (Summary, Flashcards, Mind Map)
5. Saved to CosmosDB → For retrieval and reference
6. User Exports or Interacts Further → Conversation History retained

2.3 Core Features and Functionality

The AI Videos platform has been meticulously crafted to address a wide range of content analysis needs, primarily targeting educational, professional, and accessibility-based use cases. It allows users to upload and analyze various types of media—including YouTube videos, PDFs, and image files—and transform them into structured, summarized, and engaging educational content. This chapter focuses on breaking down the core functionalities implemented during the development lifecycle, offering a detailed view of the technical implementations, user journey, and logic behind each module.

1. YouTube Video Transcript Extraction

One of the flagship features of the AI Videos platform is its ability to automatically extract transcripts from YouTube videos by simply providing a video URL.

This feature was designed to support creators, students, and researchers who want to process audio-visual content without manually transcribing it.

Feature Explanation

When a user enters a YouTube link into the system, the platform begins fetching the video metadata and looks for closed captions or auto-generated transcripts. This extraction happens in real-time, ensuring the user receives results almost instantly, provided the video is accessible and not restricted by privacy settings. The extracted transcript is cleaned, time-aligned, and saved into the system for further processing such as summarization, flashcard generation, or translation.

My Contributions

During my internship, I was directly involved in refining this feature to:

- Handle multiple language transcripts (e.g., English, Spanish, Hindi) and gracefully fall back when one is unavailable.
- Enhance error handling for restricted, unavailable, or low-quality videos.

- Enable timestamp preservation for contextual understanding.
- Add logic to switch between manual input and fetched transcript preview.
- Integrate backend API calls for transcript processing into the frontend seamlessly.

Workflow

- User Input: Video URL is entered in the system.
- Transcript Fetching: API fetches the transcript using YouTube APIs or in-house parsers.
- Cleaning & Formatting: The raw transcript is cleaned to remove noise (e.g., timestamps, breaks).
- Storage: Data is stored in a backend database with a unique slug ID.
- Display: A preview of the transcript is shown on the UI with loading animations.

Technical Highlights

- Language detection and multilingual support using Whisper model (fallback).
- Transcript segmentation for summary readiness.
- Real-time response enabled via async calls.
- Integration with CosmosDB for storing processed video data.

Benefits

- Saves hours of manual transcription effort.
- Useful for academic research, accessibility, and learning.
- Boosts content reusability for YouTubers and educators.

2. PDF Text Extraction and Summarization

Another powerful feature of the AI Videos platform is its ability to extract, summarize, and restructure text from PDF documents. This module was designed to cater to a wide range of users—from researchers and educators to corporate professionals—who often deal with large PDF reports, books, or training manuals and need to quickly extract key insights.

Feature Explanation

The PDF analysis module allows users to upload any multi-page document in PDF format. The system then parses the content, extracts raw text, and processes it using AI-powered natural language models to create condensed summaries without losing the original meaning or structure. This feature ensures that users can convert time-consuming reading material into quick, digestible insights.

The platform also supports maintaining document structure (headings, bullet points, section divisions) during extraction, which makes the summary more intuitive and context-aware.

My Contributions

As part of my core backend and frontend responsibilities, I contributed to:

- Implementing support for multi-page PDF parsing.
- Fixing extraction inconsistencies, such as repeated headers or broken text flows.
- Ensuring structure preservation using rule-based formatting and heading detection.
- Creating a responsive upload interface with real-time file validation.
- Integrating AI summarization models using prompt-tuned pipelines.
- Addressing performance issues when uploading large-sized PDFs.
- Writing utility functions to paginate long content and manage memory usage efficiently.

Workflow

1. File Upload: Users upload PDF documents via the platform's drag-and-drop interface.
2. Text Extraction: The system parses each page and extracts raw content.
3. Structure Preservation: Headings, paragraphs, and bullets are identified and retained.
4. Summarization: AI models generate summaries based on semantic clustering.
5. Preview: The summarized output is displayed in a clean, scrollable format.

Technical Highlights

- Used PDF.js and backend parsers for accurate text layer recognition.
- Introduced chunk-wise summarization to manage memory for large documents.
- Supported batch uploads and asynchronous processing.
- Maintained separation of metadata (title, author) for extended features like citation and referencing.

Benefits

- Helps students and professionals summarize research papers or reports quickly.

- Reduces cognitive load by presenting only the most essential points.
- Improves accessibility for people who prefer audio summaries (when paired with text-to-speech).

3. Image Text Extraction and Visual Description

One of the most innovative aspects of the AI Videos platform is its ability to intelligently process image-based content—a feature that brings immense value in educational, accessibility, and analytical contexts. This module allows users to upload images (in formats such as PNG, JPG, JPEG) and receive both textual data extracted from the image and a comprehensive AI-generated visual description.

This dual-layer capability bridges the gap between static visuals and dynamic understanding, enabling users to analyze posters, infographics, scanned documents, or handwritten notes with high precision and contextual awareness.

Feature Explanation

The image processing pipeline in AI Videos is powered by a combination of Optical Character Recognition (OCR) and computer vision-based captioning models. This allows the system not only to extract readable text from images but also to understand and describe what is visually present in them.

For example, if a user uploads a classroom photo or a slide with charts, the platform can:

- Read out labels and descriptions using OCR
- Generate AI-based narratives like: “A bar chart comparing revenue in 2021 and 2022 with a highlighted growth trend.”

This provides valuable metadata for users working on visual notes, online resources, or scanned research documents.

My Contributions

As an AI-driven full-stack developer, my involvement in this feature was multifaceted:

- OCR Integration: Integrated Tesseract OCR and fine-tuned parameters for high-accuracy text recognition, especially in noisy images.
- Visual Description AI Models: Integrated pre-trained models for image captioning, enabling narrative-style descriptions of uploaded visuals.
- Multi-format Compatibility: Ensured support for common image formats (JPEG, PNG, BMP).

- Responsive Interface: Designed and implemented a drag-and-drop upload panel with real-time feedback and preview capabilities.
- Batch Upload Support: Allowed processing of multiple images with individual summary outputs for each.
- Layout Optimization: Managed vertical and horizontal layouts depending on image orientation and length of output.
- Performance Tuning: Used lazy loading and asynchronous rendering to handle high-resolution images without performance lag.
- Performance Tuning: Used lazy loading and asynchronous rendering to handle high-resolution images without performance lag.

Workflow

- Upload Image: User selects or drags an image into the upload interface.
- Text Recognition: OCR is applied to detect and extract embedded text (e.g., titles, labels, numbers).
- Content Description: AI models analyze the image for layout, objects, and context to generate a human-like description.
- Display Output: Extracted text and description are displayed side by side for clarity.

Key Use Cases

- Reading content from lecture slides or handwritten notes.
- Analyzing visual assets in research or marketing reports.
- Accessibility tools for visually impaired users.
- Academic assistance for content comprehension.

Benefits

- Converts lecture images or diagrams into study-ready text.
- Enables accessibility by generating alt-text for users with visual impairments.
- Facilitates content reuse across modules like flashcards, mind maps, and summaries.

4. Flashcard and Mind Map Generation

One of the most impactful and educationally beneficial features integrated into the AI Videos platform is the automated generation of flashcards and mind maps. These tools are designed to convert long-form educational content — whether derived from YouTube videos, PDFs, or images — into compact, structured, and easily digestible learning modules.

Feature Overview

This module primarily serves students, educators, and researchers looking to reinforce knowledge through repetition and concept visualization. With just a single click, the AI system:

- Extracts key terms, facts, and relationships from the content.
- Organizes them into flashcard-style Q&A pairs.
- Builds interactive mind maps showing the logical hierarchy of ideas.

This not only promotes active recall but also strengthens conceptual understanding by visualizing the interconnections between topics.

My Contributions

As part of this feature's development and refinement, I:

- Designed the Flashcard generation logic using prompt-engineered calls to AI models for extracting definitions, examples, and terminologies.
- Implemented a mind map structuring algorithm that groups related ideas and maps parent-child relationships using hierarchical trees.
- Added export functionalities, allowing users to download flashcards as PDFs and mind maps as PNGs or interactive web widgets.
- Improved interface features like card flipping animations, theme customization, and quiz mode.

Workflow Explanation

Content Selection: After summarization, the user selects "Generate Flashcards" or "Create Mind Map."

1. Information Extraction: The system analyzes the processed text to identify:

- Definitions
- Important facts
- Hierarchical topics

2. AI Structuring:

- For Flashcards: Creates “Question” and “Answer” pairs.
- For Mind Maps: Structures data into nodes and branches based on topic clusters

3. Display & Export:

- Users view cards in flip or list mode.
- Mind map is shown interactively or as a static diagram.
- Download options are available in PDF or image format.

Technical Highlights

- Integrated Hugging Face Transformers and OpenAI’s GPT model for semantic understanding.
- Used vis.js and Mermaid.js for visual rendering of mind maps.
- Incorporated persistent state saving to CosmosDB for resuming or re-editing flashcards later.
- Optimized flashcard generation for both long text and multi-modal inputs (e.g., from images and PDFs).

Benefits to End Users

- Boosts retention: Flashcards support spaced repetition and active learning.
- Improves comprehension: Mind maps enable visual learners to grasp complex relationships.
- Saves time: Reduces manual note-making and structuring efforts.
- Highly customizable: Users can edit, rearrange, and export content.

5. Translation and Accessibility Tools

In today's global and inclusive digital landscape, ensuring that users from diverse linguistic and ability backgrounds can access and understand content is not just an added feature — it’s a necessity. The AI Videos platform tackles this challenge through its powerful Translation and Accessibility Toolkit, built to support multi-language learning and to improve accessibility for users with different abilities.

Feature Overview

The Translation and Accessibility suite consists of two key modules:

- Multilingual Translation
- Text-to-Speech (TTS) Accessibility

These tools serve a wide user base — from students learning in non-English environments to individuals with visual impairments. Together, they ensure the platform is not only globally scalable but also socially responsible in its design.

My Contributions

As part of this feature's integration, I:

- Built the language selection dropdown logic with dynamic loading of supported languages from the backend.
- Integrated APIs (Google Translate and DeepL) for accurate multi-directional translation of transcripts and summaries.
- Designed a TTS control system with features like play, pause, and speed adjustment, supporting both summaries and Q&A flashcards.
- Developed the dark mode + screen reader–friendly interface enhancements for high accessibility compliance (WCAG standards).
- Ensured that all translated content was also saved in the database under unique session keys for future retrieval.

How It Works

1. Translation Module:
    - User selects a target language.
    - The transcript, summary, flashcards, and even UI labels are translated.
    - Translations are contextually preserved, not literal, using advanced NLP.
    - Example: An English video transcript can be read and summarized in Spanish, German, or Hindi.
2. Text-to-Speech (TTS):
    - TTS buttons are available across key content sections.
    - Uses browser-native speech synthesis or integrated cloud services (Azure/CereProc).
    - Helps visually impaired users listen to summaries or question sets.

Key Functional Highlights

- Language Support: Over 25 global languages including French, Spanish, Hindi, German, Japanese, Arabic, and Mandarin.

- Auto-Detection: Detects original language of uploaded PDFs or videos and suggests translations.
- Real-time Playback: TTS plays without page refresh or re-rendering.
- Content Caching: Previously translated content loads instantly if re-selected.
- Mode Adaptation: Supports dark/light mode TTS styling, including adjustable text contrast for better visibility.

End-User Benefits

- Inclusion: Users from different countries can learn in their native language.
- Accessibility: Users with reading or visual difficulties can access information via audio.
- Speed: Instant, API-driven translations reduce manual conversion efforts.
- Education: Non-native English learners can cross-reference content between languages.

2.4 Technical Stack and Tools Used

The successful implementation of the AI Videos platform required a robust, scalable, and modular technology stack. The complexity of handling diverse media types (videos, PDFs, images), along with real-time AI processing, multilingual support, and a responsive interface, demanded the careful selection of technologies that offer high performance, seamless integration, and ease of maintenance.

1. Frontend Technologies

The frontend was designed to be visually appealing, user-friendly, and responsive across devices. Emphasis was placed on interaction design, accessibility, and dynamic content rendering.

- React.js: The primary JavaScript library used for building the user interface. React offered component reusability, virtual DOM performance, and a vibrant ecosystem of libraries.
- Tailwind CSS: Utilized for designing a modern, utility-first responsive UI. Allowed quick styling of components with custom themes for dark and light modes.
- JavaScript (ES6+): Used extensively for dynamic event handling, form validation, client-side logic, and animations.

- Axios: A promise-based HTTP client used to communicate with backend APIs and fetch data in real-time.

2. AI & NLP Technologies

Since the platform's core revolved around AI-based summarization and analysis, several machine learning models and services were integrated:

- Google Gemini/OpenAI APIs: Used for advanced summarization, content generation, and language model operations.
- Whisper (by OpenAI): Deployed for speech-to-text transcription of YouTube videos when closed captions were unavailable.
- Tesseract.js: JavaScript OCR engine used for extracting text from image content.
- spaCy / NLTK (Python): For backend NLP tasks like keyword extraction, flashcard generation, and sentence parsing.
- json2video: Utilized in future upgrades to convert structured AI outputs (like transcripts or summaries) into animated videos with captions.

3. Backend Technologies

A high-performance backend was crucial to handle media uploads, processing requests, and communication with AI engines.

- Node.js + Express.js: Used for server-side logic, routing, and managing API endpoints.
- Python (Flask/Middleware): Deployed selectively for specific AI/NLP functionalities not supported natively in Node.js.
- FFmpeg: Command-line tool used for extracting audio from YouTube videos for speech-to-text conversion and timestamp alignment.
- Multer: Used for handling file uploads (PDFs, images) from users securely.

4. Database and Storage

To manage persistent history, session data, user queries, and outputs, the following storage solutions were used:

- CosmosDB (Azure): A globally distributed, multi-model NoSQL database used to store chat histories, session details, and output logs.
- MongoDB (Testing Phase): Initially used for local development due to schema

flexibility and JSON-based querying.

- Cloudinary: Temporarily used for hosting user-uploaded media files during processing.

5. Cloud & Deployment Tools

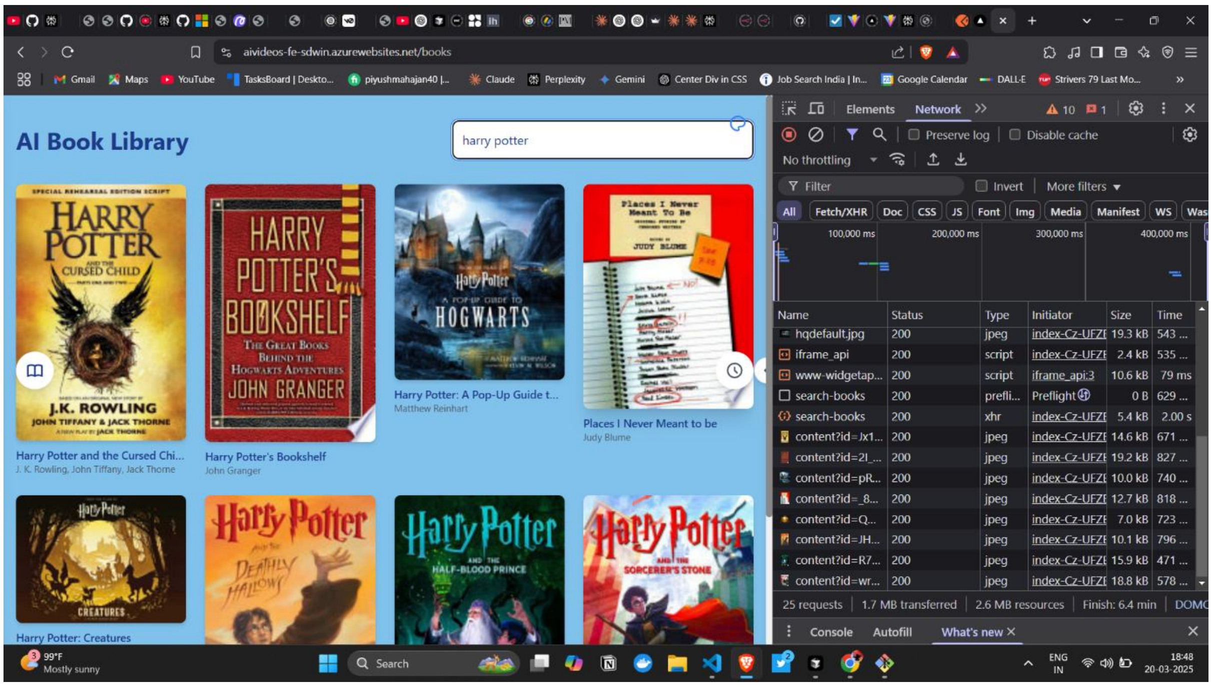
- Microsoft Azure: Used for hosting the platform, deploying APIs, and running backend workers. Azure App Services and CosmosDB were pivotal.
- Vercel: Used for frontend hosting during development and testing stages.
- Docker: Employed to containerize key components, allowing consistent deployment across environments.
- Git & GitHub: For version control, issue tracking, and collaborative development.

6. Testing & Debugging Tools

- Postman: Used to test backend APIs and check responses during development.
- Jest + React Testing Library: Implemented to test frontend component behavior and interaction.
- Browser DevTools: Extensively used for debugging of UI layouts and JS execution.
- VS Code Extensions: Tools like Prettier, ESLint, and GitLens improved development quality and formatting.

2.5 My Contributions and Development Work

As a Full-Time AI-Driven Full Stack Developer at Tangible IT, Mexico, my role extended far beyond traditional coding assignments. I was involved in the entire lifecycle of the AI Videos platform—from planning and design, to feature implementation, optimization, deployment, and testing. The goal was not just to build isolated components but to contribute to a unified, production-ready system that could handle complex media types while maintaining a user-centric, accessible interface.



Name	Status	Type	Initiator	Size	Time
hqdefault.jpg	200	jpeg	index-Cz-UFZE	19.3 kB	543 ...
iframe_api	200	script	index-Cz-UFZE	2.4 kB	535 ...
www-widgetap...	200	script	iframe_api:3	10.6 kB	79 ms
search-books	200	prefli...	Preflight	0 B	629 ...
search-books	200	xhr	index-Cz-UFZE	5.4 kB	2.00 s
content?id=Jx1...	200	jpeg	index-Cz-UFZE	14.6 kB	671 ...
content?id=2I_...	200	jpeg	index-Cz-UFZE	19.2 kB	827 ...
content?id=pR...	200	jpeg	index-Cz-UFZE	10.0 kB	740 ...
content?id=_8...	200	jpeg	index-Cz-UFZE	12.7 kB	818 ...
content?id=Q...	200	jpeg	index-Cz-UFZE	7.0 kB	723 ...
content?id=JH...	200	jpeg	index-Cz-UFZE	10.1 kB	796 ...
content?id=R7...	200	jpeg	index-Cz-UFZE	15.9 kB	471 ...
content?id=wr...	200	jpeg	index-Cz-UFZE	18.8 kB	578 ...

25 requests | 1.7 MB transferred | 2.6 MB resources | Finish: 6.4 min

Fig No. 2.3: AI Book Library Feature

My responsibilities spanned across both frontend and backend, as well as AI integration, state management, performance optimization, and collaborative workflows with the product and engineering teams. Below is a detailed breakdown of my work.

UI/UX Development and Enhancement

- Developed responsive and modular components using React.js and Tailwind CSS, ensuring adaptability across various devices and screen sizes.
- Integrated theme-switching logic for light/dark modes, enhancing accessibility for diverse user preferences.
- Designed custom input-type tabs (YouTube, PDF, Image) with intelligent state control, ensuring outputs were reset appropriately to avoid confusion during content switching.
- Worked on form validation, loading animations, and interactive transitions that improved the overall user experience.

Backend & Middleware Integration

- Created and optimized Express.js APIs to interact with the AI processing layer, ensuring smooth data flow between the frontend and backend.
- Used Multer and FFmpeg to process media files—extracting audio, managing uploads, and converting formats for analysis.

- Integrated Python middleware (via Flask) to connect Whisper, Tesseract, and Gemini APIs for transcription and content generation.
- Proposed and implemented logic for clearing previous outputs and resetting views to the "Summary" tab on each new submission, solving a common UX confusion.

CosmosDB Schema Fixes and Session Management

- Resolved persistent history issues by redesigning the schema for CosmosDB to store data using slug, title, and user session ID.
- Added logic to dynamically generate conversation titles based on the AI output or fallback to the user's input.
- Debugged and fixed issues in the slug creation logic, ensuring each session was uniquely identifiable and retrievable via URL.

AI and NLP Feature Integration

- Integrated Whisper API for extracting transcripts from YouTube videos without captions.
- Implemented Gemini AI and OpenAI models for content summarization, Q&A generation, flashcards, and mind maps.
- Enhanced multilingual translation features and enabled text-to-speech output for improved accessibility.
- Ensured cross-model state persistence for seamless interactions across content types.

Documentation and Workflow Optimization

- Authored detailed internal documentation covering component responsibilities, API usage, deployment instructions, and testing procedures.
- Participated in daily stand-ups and code reviews, where I implemented feedback loops to enhance code readability and performance.

2.6 Challenges Faced and Resolutions

Every major software project presents a spectrum of challenges — technical, collaborative, and conceptual. While building the AI Videos platform, I encountered several obstacles that tested not only my programming knowledge but also my problem-solving, communication, and adaptability. This section outlines the major challenges I faced during development and the strategic resolutions applied to overcome them.

1. Cosmos DB Integration Issues

Challenge:

Initially, the chat history and conversation state were not being saved or retrieved correctly. CosmosDB documents were either overwritten or stored without proper indexing, making the session retrieval inconsistent. It also affected the unique slug-based routing feature.

Resolution:

- I redesigned the schema structure to include slug, title, timestamp, and user session ID, ensuring each session could be tracked uniquely.
- Added fail-safe logic to regenerate a slug if a duplicate was detected.
- Implemented a dynamic title generator using AI output or user input fallback, making history browsing user-friendly.

2. Output Overlap Between Input Types

Challenge:

When users switched between YouTube, PDF, and Image input modes, old outputs (summaries, flashcards, etc.) would persist, creating confusion and UI clutter.

Resolution:

- Introduced state-reset hooks that clear previous data when switching input types.
- Ensured the system always returned to the default Summary tab after a new file or link was submitted.
- Used conditional rendering to hide output sections until new content was loaded.

3. Schema Conflicts During Morphic Integration

Challenge:

Morphic and AI Videos had different data schemas, leading to failures in shared components and API compatibility errors during development.

Resolution:

- Conducted a schema normalization audit to align data formats and keys.
- Created middleware adapters that transformed payloads dynamically based on the source module.
- Added error logging to highlight mismatched fields and ensure debugging was efficient.

4. AI Model Latency and Performance

Challenge:

During transcript summarization or flashcard generation, the Whisper and Gemini models sometimes took too long to respond, affecting user experience.

Resolution:

- Implemented asynchronous feedback indicators (loading bars, disabled buttons) to signal active processing.
- Added AI model timeout handling, returning cached partial results or suggesting smaller content inputs if delays exceeded threshold.
- Proposed and later integrated lazy loading to load result sections progressively.

2.7 Testing and Deployment Process

The AI Videos application underwent a structured testing and deployment process to ensure a stable and efficient experience for end users. As part of the development team, I actively contributed to both manual and automated testing cycles.

Testing Strategy:

- Unit Testing: Core utility functions and API handlers were tested using console-based test cases to ensure correct functionality.
- Integration Testing: API responses and UI interactions were verified through Postman and browser-based testing to check end-to-end workflows.
- User Interface Testing: Multiple layout scenarios (dark/light mode, language toggles, input switching) were manually validated for responsiveness and visual consistency.

Deployment Approach:

- All new features were first pushed to the pre-production environment to undergo staging validation.
- Once approved, they were deployed to the production environment with close monitoring for performance and reliability.
- Azure DevOps pipelines and manual version checks ensured rollback safety and version control.

This phase ensured that the application remained stable, responsive, and secure throughout active development

Chapter 3

AI DISCOVER PROJECT

In the modern digital era, where information is abundant but scattered, the need for an intelligent, real-time, and context-aware assistant has become more crucial than ever. Users—ranging from students and educators to corporate professionals—require solutions that can go beyond static search results and offer dynamic, AI-driven insights tailored to their queries. Recognizing this gap, the AI Discover platform was conceptualized and developed as a next-generation conversational AI assistant, designed to transform the way we interact with web-based knowledge and natural language interfaces.

3.1 Introduction

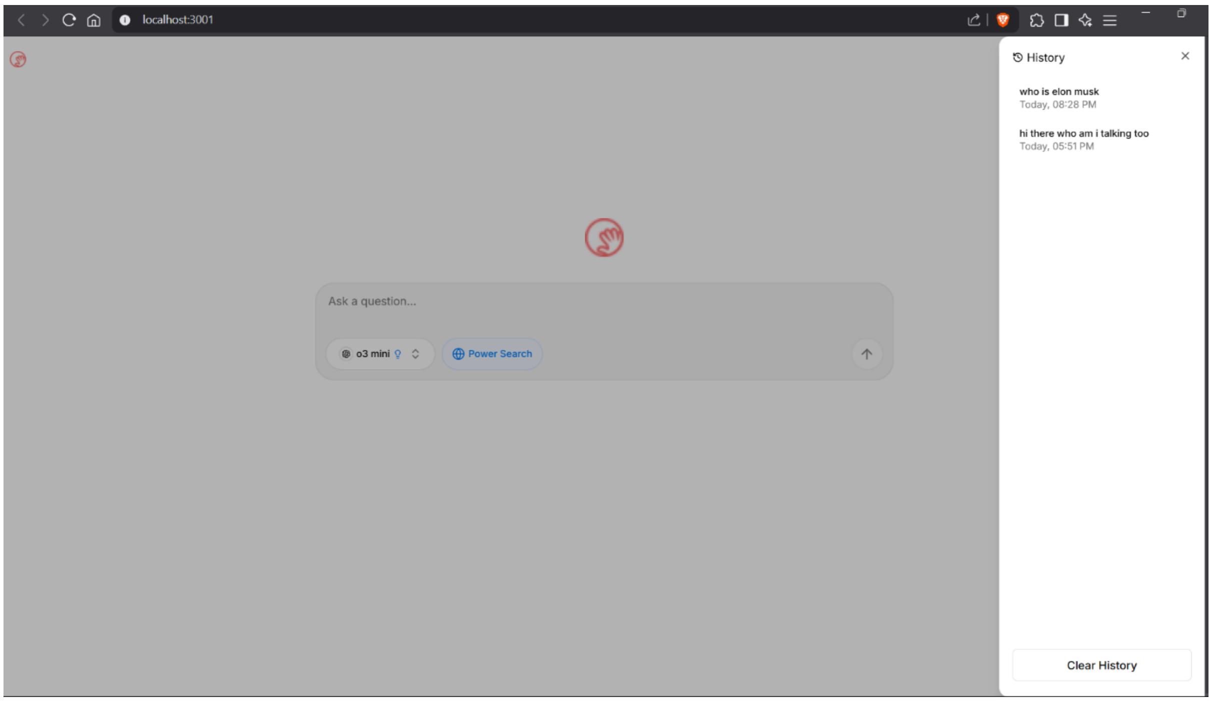
AI Discover is not a conventional chatbot; it is an integrated, intelligent search and conversation system built to deliver fact-based, real-time, multi-modal responses. It leverages Large Language Models (LLMs), enhanced with real-time web search capabilities, image rendering tools, and conversation persistence mechanisms to offer a holistic interaction experience. Its intuitive interface enables users to initiate a conversation, ask diverse questions, and receive responses that are not only textually rich but also visually informative.

Unlike typical bots that rely on pre-trained data or a restricted knowledge base, AI Discover is equipped to fetch fresh web content, validate it, and present it in an organized format. It also identifies when a query requires image-based results, seamlessly integrating relevant visuals using lazy loading techniques and responsive design components. This adds a layer of interactivity and personalization, significantly improving user engagement.

From a technological perspective, AI Discover is an enterprise-grade, full-stack solution, integrating a React-based frontend, Node.js backend, Azure-based deployment pipelines, and CosmosDB for real-time storage. It is engineered to handle both authenticated and anonymous users, offering differentiated functionalities such as full chat history deletion, session-based conversation loading, and intelligent message role categorization. This makes it scalable, secure, and adaptable for use in both standalone applications and embedded environments like mobile learning platforms.

One of the most defining features of AI Discover is its context retention capability. The application supports multi-turn conversations, maintaining memory of past exchanges and intelligently adapting future responses based on conversation history. Whether a user is discussing climate change, artificial intelligence, or medical research, the system retains contextual depth, thus offering a coherent and human-like experience across prolonged sessions.

The project also emphasized UI/UX excellence—ranging from visually enriched chat layouts to dynamic sidebar histories and sleek loading feedback elements. Conversations are automatically titled using AI-driven logic, categorized based on user roles (user/tool/assistant), and can be restored using unique identifiers such as /chat/[id] in the URL.



who is elon musk
Today, 08:28 PM

hi there who am i talking too
Today, 05:51 PM

Fig No. 3.1: AI Discover Dashboard History

AI Discover interface showing the chat input and conversation history panel with titled messages and timestamps.

In essence, AI Discover was envisioned and implemented as a digital bridge between human curiosity and structured knowledge, combining the strengths of language modeling, search intelligence, and cloud infrastructure. It empowers users not just to receive answers, but to engage in a fluid, intelligent conversation with a machine that understands, responds, and evolves with their inputs.

This chapter explores the entire journey of AI Discover—from its core objectives and system architecture to feature development, challenges faced, tools utilized, and the final deployment process. Through this project, I gained invaluable experience in integrating real-world APIs, managing stateful user interactions, and building a secure, interactive system that redefines what an AI assistant can do.

3.2 Objectives of AI Discover Platform

The development of the AI Discover platform was guided by a clear vision: to create a context-aware, real-time AI assistant that goes beyond traditional chatbots and delivers verifiable, intelligent, and visually enriched responses to user queries.

Built with the goal of bridging the gap between conversational AI and real-time information retrieval, the platform was designed to empower users by enabling natural language interaction with dynamic web data, while retaining contextual memory, user history, and cross-tool functionality.

This section elaborates on the key objectives that formed the foundation of the AI Discover system.

1. Deliver Real-Time, Search-Backed Answers

A primary objective of AI Discover was to act as an intelligent conversational layer over real-time web data. Unlike static AI models that rely solely on pre-trained knowledge, AI Discover is integrated with web search tools that enable it to:

- Perform live web lookups.
- Retrieve the most recent and relevant content.
- Process search results and generate AI-summarized responses.

This ensures that user queries—whether about breaking news, emerging technologies, or public figures—receive fact-checked, up-to-date answers with embedded source references.

2. Support Multi-Turn, Context-Aware Conversations

The system was designed to handle multi-turn dialogue, where the assistant could maintain and understand previous user inputs, even across extended sessions. This objective required building:

- A persistent memory layer backed by CosmosDB.
- Dynamic role tracking (user/tool/assistant).
- Thread-aware response logic that adapts based on conversation history.

This conversational depth enabled users to ask follow-up questions or pivot topics without losing the narrative flow—a hallmark of human-like interaction.

3. Integrate Visual Intelligence through Image Query Handling

AI Discover aimed to be more than a text-based platform. A core goal was to understand and respond to image-specific queries by:

- Automatically detecting when a user asks for an image.

- Triggering an image search tool in the backend.
- Displaying results with responsive UI components, including rounded image cards, lazy loading, and visual effects.

This feature added a multi-modal dimension to the platform, making it suitable for users who needed visual references along with text.

4. Provide Persistent Conversation History and Session Management

Another strategic objective was to implement a history-aware assistant where:

- Each conversation was saved with a unique identifier (/chat/:id).
- AI-generated titles were automatically assigned based on the conversation's context or the assistant's reply.
- Users could revisit, delete, or continue previous conversations.

This feature was especially useful in educational and professional use cases, where users may wish to revisit a prior research session or refer back to AI-generated insights.

5. Handle Role-Based Functionality and Secure Data Flows

AI Discover was designed to treat anonymous and authenticated users differently, ensuring both usability and privacy:

- Authenticated users could delete all messages in a session, rename threads, and sync conversations across devices.
- Anonymous users had restricted permissions (e.g., delete message-by-message), preserving privacy without login requirements.

This role-based handling helped strike a balance between security, personalization, and open access.

6. Adapt to Application Contexts Dynamically

A key technical objective was to allow AI Discover to detect the environment it was launched in and adapt its functionality accordingly. For example:

- If the user accessed it from the ACV App, a query flag like isFromACV=true adjusted the toolset, UI, and logging behaviors.
- This flexibility allowed AI Discover to act as a plug-and-play assistant module for integration with other enterprise platforms.

7. Optimize User Experience with Feedback and Performance Enhancements

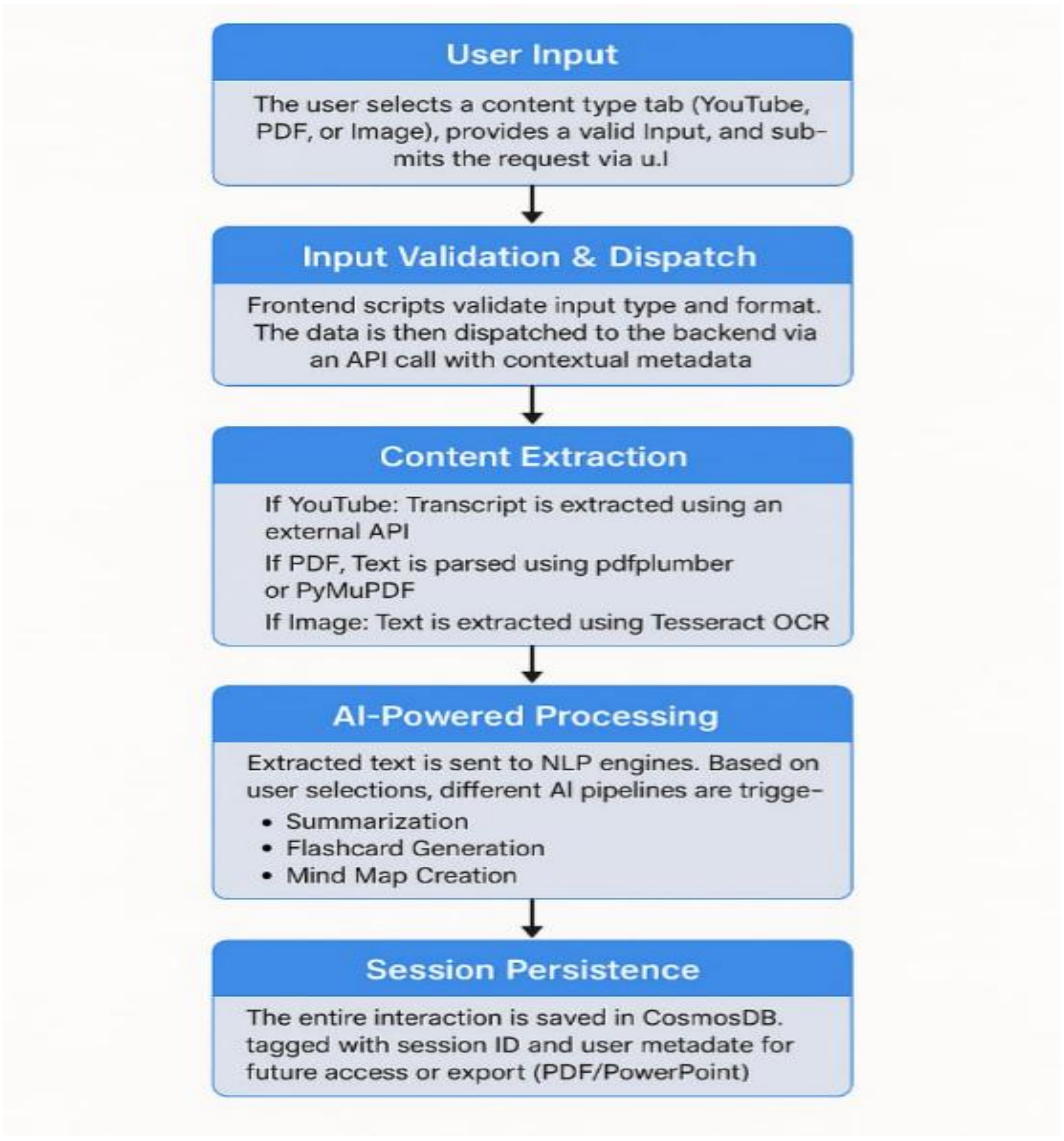
To ensure a smooth and interactive experience, the platform included:

- Animated loading states to indicate processing.
- Fallback messaging when external APIs timed out.
- UI elements like sidebars, floating buttons, and toast alerts to improve navigation.

These UX design decisions were driven by the objective to keep users informed, engaged, and confident in the system's behavior at all times.

3.3 System Architecture and Workflow

The architecture of AI Discover is designed as a modular, full-stack system, ensuring real-time interaction, state persistence, and intelligent response generation. It comprises several tightly integrated components—from user interface and backend services to AI processing pipelines and cloud-hosted databases. The workflow supports both stateless query processing and stateful conversational management, depending on whether the user is anonymous or logged in.



User Input
The user selects a content type tab (YouTube, PDF, or Image), provides a valid Input, and sub-mits the request via u.I
Input Validation & Dispatch
Frontend scripts validate input type and format. The data is then dispatched to the backend via an API call with contextual metadata
Content Extraction
If YouTube: Transcript is extracted using an external API
If PDF, Text is parsed using pdfplumber or PyMuPDF
If Image: Text is extracted using Tesseract OCR
AI-Powered Processing
Extracted text is sent to NLP engines. Based on user selections, different AI pipelines are trigge-
• Summarization
• Flashcard Generation
• Mind Map Creation
Session Persistence
The entire interaction is saved in CosmosDB. tagged with session ID and user metadate for future access or export (PDF/PowerPoint)

Fig No. 3.2: System Architecture Overview AI Discover

The diagram above illustrates the complete processing pipeline of the system, starting from user input to final session storage. Each stage represents a key functional module—ranging from content validation and extraction to AI-driven analysis and persistent data handling—ensuring efficient, accurate, and user-personalized output.

Frontend Layer

The frontend of AI Discover is built using React.js, enabling a smooth, responsive, and component-driven interface that adapts based on screen size, user role, and query type.

Key Functionalities:

- Input Bar with support for voice/text inputs and search triggers.
- Sidebar with History Panel for persistent session access and dynamic routing (/chat/:id).
- Image Result Renderer that adjusts based on query context.
- Theme Management with light/dark toggle and high contrast for accessibility.

Backend and AI Integration

The backend is developed using Node.js with Express, managing:

- Web routing
- Query processing
- Tool activation (e.g., search or image)
- Communication with AI engines

The platform uses a tool-first approach:

- A "tool" fetches the data (like web search or image result),
- Then the "assistant" (LLM) summarizes or interprets it.

AI Engines Used:

- OpenAI GPT / Gemini – For summarizing and answering based on the tool's output.
- Search Tool – Triggers web search and formats URLs/content for AI.
- Image Tool – Detects when image queries are made and returns visual results.

Database & Session Management

CosmosDB, a cloud-native NoSQL database, is used to manage:

- User queries and assistant replies
- Tool responses (web content, image links)
- Auto-generated conversation titles
- Unique session IDs and routing paths

All conversation threads are stored using structured roles:

- user
- tool

- assistant

Image Fetching and Display Logic

When a user asks for an image (e.g., “Show me AI robot designs”), the system:

1. Classifies the query as image-related.
2. Activates the image tool to fetch relevant images.
3. Stores image results with meta-data (alt, source, ID).
4. Displays them using a responsive card layout with shadow and hover effects.

The logic is built to lazy-load images, reducing page lag, and enhancing performance on low-bandwidth networks.

Context Handling and Role-Based Logic

AI Discover is context-aware. It tracks:

- Which user is active (authenticated or guest)
- What tool was last used
- What the assistant’s last reply was

Role logic is used to:

- Display clear differentiation in the UI (user/tool/assistant messages).
- Dynamically route the user to the right /chat/:id view.
- Allow or restrict features like “delete entire history” for guests vs. logged-in users.

Features Enabled by Role Logic:

- Intelligent title generation
- Per-message vs. full-chat deletion
- Cross-tab session persistence

3.4 Key Features and Functionality

The core strength of AI Discover lies in its feature-rich architecture that delivers a responsive, intelligent, and user-centric chat experience. From real-time web integration to structured conversation management and multimedia content rendering, each functionality was meticulously designed to elevate the traditional chatbot into a true AI-powered assistant.

This section highlights the major features built into the system, categorized by their technical purpose and user impact.

Real-Time Web Search Integration

AI Discover is equipped with a "Tool + Assistant" pipeline where user queries are first handled by a web search tool. This component:

- Fetches current information directly from the web.
- Parses relevant snippets and URLs.
- Sends structured data to the language model for summarization.

This ensures the assistant's answers are based on live content, making it ideal for trending queries, updated statistics, and current affairs.

Natural Language Q&A Interface

The platform supports free-form conversational queries, such as:

- "Who is the current CEO of OpenAI?"
- "Summarize the purpose of quantum computing."
- "Give me five recent advances in space tech."

Thanks to context awareness, users can follow up without restating their entire question. The assistant understands past references, topic continuity, and intent.

Image Detection and Visual Response Handling

One of the standout features is its ability to:

- Recognize when a user's query is image-based.
- Activate an image search API.
- Display results in a responsive, card-based gallery with:
    - Rounded corners
    - Shadow effects
    - Lazy loading

Users can visually interact with results without needing to leave the chat interface.

Intelligent History Panel

Every conversation is:

- Saved in CosmosDB.
- Automatically titled (based on AI output or user input).
- Assigned a unique route like /chat/abc123.

The sidebar lets users:

- View past sessions.
- Click to reload full conversations.
- Delete individual or all sessions (based on authentication level).

This enhances usability by offering conversational continuity, even after logout or device change.

Role-Based Access and Permissions

The system identifies each user's session status:

- **Anonymous users**: Can delete individual messages only.
- **Authenticated users**: Can delete entire chat threads, sync across devices, and export history.

Role-based logic also governs:

- Sidebar visibility
- Assistant permissions
- Secure history management

Intelligent Title Generation

Using either the first assistant replies or the user's question, the system generates meaningful titles like:

- "AI and Space Exploration"
- "What is Cybersecurity?"

This helps users locate past conversations easily and improves session organization.

Error Handling, Feedback, and UI Responsiveness

The platform includes:

- Loading indicators for queries in progress
- Tool/assistant failure fallbacks with retry options
- Toast messages and state transitions for better UX

It ensures that the user is never left confused or uncertain during delays or backend hiccups.

Context-Sensitive Deployment Mode (ACV App Integration)

A special flag (isFromACV) allows AI Discover to:

- Adapt when embedded inside another application (like the ACV learning app).
- Modify tools, layout, or behavior based on that context.
- Store specific context flags in the backend for session-specific adjustments.

3.5 Technical Stack and Tools Used

The successful development and deployment of AI Discover relied heavily on a carefully selected set of technologies. The project required real-time interaction capabilities, intelligent language model integration, image and data fetching, stateful session handling, and a seamless user experience. To meet these demands, the platform was built using a modern full-stack architecture, combining powerful frontend frameworks, backend APIs, cloud-based databases, and advanced AI services.

This section outlines the complete technology stack used to build the platform across five core layers.

1. Frontend Technologies

The frontend was developed with a focus on responsiveness, modularity, and user interactivity.

React.js

Served as the core JavaScript library to build a dynamic, component-based user interface. It handled routing, rendering, and component communication.

Tailwind CSS

Utilized for utility-first, responsive styling. Tailwind enabled rapid design adjustments, particularly for dark mode, button transitions, and custom modals.

React Icons & Framer Motion

Used to enhance UI aesthetics with icons and smooth component animations.

Axios

Responsible for communicating with the backend API for all tool, assistant, and history-related operations.

Responsive Web Design (RWD)

Ensured that the application was fully functional on desktops, tablets, and mobile devices without loss of layout or features.

2. Backend Technologies

The backend logic was handled using scalable, asynchronous systems capable of integrating external APIs and large language models.

Node.js + Express.js

Built the RESTful API endpoints to:

- Accept user queries
- Route queries to the appropriate tools
- Pass results to the assistant (LLM)
- Return structured responses

OpenAI / Gemini API

Used for text generation and summarization based on tool output. Powered the assistant's intelligence.

Image Search Tool (Custom API + Crawler)

Activated during image-based queries. It fetched and returned the top visual results in JSON format for rendering.

Middleware Layer

Included context evaluation functions, query formatters, and output validators before final dispatch.

3. Database & Storage

To manage persistent conversations, user sessions, tool results, and message metadata, a cloud NoSQL database was used:

Azure CosmosDB Offered:

- Schema-flexible document storage
- Session-based querying using slug and email
- Role-based data categorization (user/tool/assistant)

Blob Storage (Azure)

Temporarily used for media file caching and future scalability testing.

4. Deployment and Cloud Infrastructure

The application was hosted on a robust, scalable environment using modern DevOps tools and cloud services:

- Microsoft Azure App Services: Hosted the backend logic and APIs, offering autoscaling and global redundancy.
- Vercel: Used for frontend hosting and deployment during development and pre-production stages.
- CI/CD Pipeline (GitHub Actions): Automated testing, linting, and deployment every time a new feature or patch was committed.
- Domain Routing and SSL: Implemented via Vercel and Azure integration for production-grade security.

3.6 My Contributions and Development Work

Working on the AI Discover platform as a Full-Time AI-Driven Full Stack Developer was a dynamic and deeply enriching experience. My responsibilities spanned the full spectrum of software development—from designing key user interface components to optimizing backend workflows and resolving database integration challenges. This hands-on experience significantly elevated my technical expertise, problem-solving ability, and understanding of collaborative development in a production-grade environment.

The following detailed breakdown highlights my major contributions across different phases of the AI Discover platform:

1. Frontend Development

I contributed extensively to the frontend interface, ensuring a seamless, visually appealing, and accessible user experience.

- **Chat Interface Enhancements:**
  - Implemented the structured chat layout that displays the user's input, tool-generated output (web search or image), and the assistant's summarized response in a clean sequence.
  - Enabled role-based message styling for clear differentiation between user, tool, and assistant replies.
- **Responsive Sidebar (History Panel):**
  - Built a dynamic history panel to list past conversations.
  - Implemented intelligent title generation logic—defaulting to the assistant's response or fallback user input.
  - Managed route-based rendering (/chat/[id]) for each chat using React Router.
- **Image Display Modules:**
  - Integrated rounded, responsive image cards with hover effects and lazy loading.
  - Handled fallback behavior for failed image loads gracefully.
- **Stateful UI Handling:**
  - Managed tab transitions, content loading indicators, and dark/light mode toggles.
  - Applied logic to ensure consistent reset of views when switching queries or tools.

2. Backend Logic and Integration

The backend required robust API coordination, logic segregation, and data flow orchestration—key areas where I delivered substantial contributions:

- **API Call Optimization:**
  - Refactored and modularized backend functions for tool queries and LLM summarization.
  - Reduced redundant API calls using memoization and cleanup logic.
- **Tool-to-Assistant Pipeline:**
  - Built a logic chain where user input goes to tools (web/image search), which then feeds responses to the assistant for summarization.
  - Ensured smooth communication using async/await functions and Axios

pipelines.

- **Image Tool Enhancements:**
    - Enhanced image detection logic for image-specific queries, triggering the appropriate tool.
    - Implemented server-side image validation and response formatting.
- **Performance Handling:**
    - Added graceful degradation for tools with fallback messages in case of timeouts or empty results.
    - Developed intelligent retry logic to minimize disruptions in user interaction.

3. Database Management (CosmosDB)

CosmosDB played a central role in storing, retrieving, and managing user session data.

- **Persistent Session Handling:**
    - Designed and implemented the schema for storing messages with properties such as chatId, slug, email, timestamp, and role.
    - Supported anonymous and authenticated sessions with conditional access logic.
- **Slug and Email Filtering:**
    - Added filters to retrieve messages using user email or slug, based on the session context.
    - Enabled conversation continuation through saved history.
- **Conversation Deletion Logic:**
    - Developed delete options at both granular and global levels.
    - Ensured that anonymous users could only delete individual conversations.

4. Session Context and Parameter Handling

To customize the application behavior for different user entry points, I implemented logic for parameter-based customization.

- **isFromACV Parameter Detection:**
    - Built detection logic for isFromACV to toggle interface elements, access controls, or restrict features when the app was accessed from within the ACV ecosystem.

5. Documentation and Debugging

Beyond development, I contributed to the maintainability of the platform through well-structured documentation and proactive debugging.

- **Internal Developer Documentation:**

    - Drafted detailed internal docs outlining API usage, conversation schema structure, and component relationships.
    - Shared it with future team members to ease onboarding and iteration.
- **Bug Resolution:**
    - Investigated and resolved issues related to image tool output rendering, tab stat mismanagement, and data persistence anomalies.
    - Added error boundaries to critical frontend components.

Chapter 4

TOOLS AND TECHNOLOGIES USED

In the modern software development lifecycle, the selection of tools, frameworks, and development environments plays a vital role in ensuring efficiency, scalability, and maintainability. Throughout my internship at Tangible IT, Mexico, as a Full-Time AI-Driven Full Stack Developer, I had the opportunity to work with a variety of cutting-edge tools and technologies across both frontend and backend domains. The two major platforms I contributed to — AI Videos and AI Discover — were developed using modern JavaScript frameworks, AI integration libraries, cloud-based databases, and robust deployment pipelines.

This chapter provides a comprehensive overview of the technical stack used throughout my internship. It elaborates on the tools chosen for frontend development, backend logic, database modeling, cloud services, version control, and team collaboration. Understanding this toolchain not only enhanced my technical skill set but also helped me appreciate the significance of tool compatibility, environment configuration, and cross-platform deployment strategies in real-world production systems.

4.1 Overview of Tools and Technologies

Throughout my full-time internship at Tangible IT, Mexico, the development of enterprise-grade AI platforms like AI Videos and AI Discover demanded the use of a broad and sophisticated spectrum of technologies. These tools were not just limited to conventional frontend and backend development but also spanned across domains such as artificial intelligence, real-time API integration, database optimization, cloud hosting, version control, and cross-functional team collaboration.

The complexity of the platforms required a technology stack that was both modular and scalable. This included state-of-the-art frontend libraries for UI development, asynchronous communication systems for API orchestration, NoSQL cloud-based databases for real-time history storage, and AI/ML libraries for processing large volumes of unstructured content from diverse sources like YouTube videos, PDFs, and images. Each tool was selected for its specific performance in terms of speed, flexibility, support for large-scale data, and developer community strength.

On the frontend, React.js was used as the core library, thanks to its component-based architecture and efficient rendering via the virtual DOM. For styling and responsive design,

we used Tailwind CSS, which enabled rapid UI development with utility-first classes.
To enhance interactivity and accessibility, components like modals, tooltips, and tab transitions were powered by custom hooks and context providers.
For backend development, Node.js served as the runtime environment, with Express.js handling routing and middleware logic. These were chosen for their asynchronous, non-blocking nature, which is ideal for handling multiple API requests in real time. MongoDB was initially explored, but the final database solution for conversation history and chat logs was Azure CosmosDB, due to its native integration with Microsoft's cloud ecosystem and support for scalable, multi-region data storage.
AI functionalities like summarization, translation, flashcard generation, and mind map creation were implemented using a combination of OpenAI's language models, NLP pipelines, and custom-trained tools. External APIs were also incorporated, including Google Search API for real-time web scraping in AI Discover and YouTube Transcript APIs for video content parsing in AI Videos.
To manage development lifecycle processes, Git and GitHub were used extensively for version control, feature branching, and code reviews. All projects followed an agile workflow using tools like Notion and Jira to manage tasks, assign tickets, and maintain sprint velocity. Deployment pipelines were set up using Azure DevOps for backend services and Vercel for frontend hosting, offering seamless integration and rollback mechanisms in case of version failures.
The diversity and depth of these technologies not only supported high-performance application delivery but also provided me with real-world experience in managing complex, large-scale systems. This technical exposure reinforced my understanding of choosing the right tool for the right job — an essential competency in modern full-stack development.

4.2 Technologies Used

During the development of both AI Videos and AI Discover, a crucial part of the engineering strategy was the selection and effective implementation of modern development frameworks and libraries. These frameworks not only accelerated the development process but also helped ensure scalability, maintainability, and high-performance delivery of the platforms in real-world conditions.

Frontend Framework – React.js

At the heart of both user interfaces lies React.js, a powerful JavaScript library for building

dynamic and interactive UI components. React's virtual DOM and component-based architecture made it ideal for developing responsive, modular user interfaces where state and props could be easily managed and passed across components.
React allowed us to implement interactive elements such as:

- Chat windows with live message updates
- Dynamic rendering of user queries, assistant responses, and web tool outputs
- State-preserving tabs for switching between inputs (video, PDF, image)
- Dark mode toggles and session-based UI management

The use of React Hooks such as useState, useEffect, useRef, and useContext enabled us to manage complex logic across chat history, asynchronous responses, input state, and session persistence without polluting the global state.

Styling – Tailwind CSS and CSS Modules

For consistent design and styling, we employed Tailwind CSS, a utility-first framework that allowed for rapid prototyping and theming. Tailwind ensured visual consistency across the app, responsive layouts across devices, and performance optimizations by purging unused styles in production builds.
In parts where modular styling was required—especially in legacy UI elements or deeply nested components—CSS Modules were used to isolate styles and avoid conflicts, giving us the best of both utility and scoped CSS.

Backend Framework – Node.js with Express.js

The server-side logic was developed using Node.js, selected for its high concurrency and event-driven architecture, which made it ideal for handling multiple API calls, managing real-time input-output flows, and ensuring non-blocking operations.
On top of Node, Express.js provided a lightweight and unopinionated web framework. Express helped us:

- Define modular REST APIs for summarization, image analysis, PDF parsing, and translation
- Handle routes for CRUD operations on CosmosDB
- Manage authentication, session tracking, and user query logging
- Integrate third-party tools and middleware seamlessly

Database Management – Azure CosmosDB

While MongoDB was initially used in prototype stages, production readiness required a cloud-native, horizontally scalable solution. Azure CosmosDB was selected due to:

- Global distribution and low-latency read/writes

- Native JSON document support
- Easy integration with Microsoft-based deployment pipelines

CosmosDB supported our requirement for persistent chat history, user role-based data retrieval, and slug-based session titles.

AI Libraries – OpenAI, Whisper, Tesseract.js

AI Videos and AI Discover integrated multiple AI libraries and models:

- OpenAI GPT APIs were used for summarization, Q&A, and content transformation
- Whisper by OpenAI supported audio-to-text conversion for YouTube videos with no transcript
- Tesseract.js, a JavaScript OCR engine, was used for image-to-text extraction in image analysis
- LangChain and Hugging Face Transformers (experimental) were explored for advanced NLP tasks

Deployment & DevOps Tools – Vercel, Azure, Git, GitHub

- Frontend hosting was managed on Vercel, chosen for its zero-config CI/CD pipeline, GitHub integration, and preview deployments for each pull request.
- Backend APIs were deployed on Azure Functions and Azure App Services for scalability.
- Version Control was maintained through Git, with structured commit messages and pull request reviews on GitHub to track feature additions, bug fixes, and refactorings.
- DevOps activities also included environment separation (staging, pre-production, production) to ensure reliability.

4.3 Software and Productivity Tools

In a fast-paced, collaborative development environment like **Tangible IT**, the use of robust software and productivity tools is as critical as coding itself. These tools empower developers to work efficiently, maintain code quality, manage tasks, facilitate communication, and ensure that the product life cycle moves forward without unnecessary friction. Throughout my internship, I actively used a wide variety of tools that significantly enhanced both individual productivity and team collaboration.

1. Visual Studio Code (VS Code)

Its vast ecosystem of extensions allowed me to:

- Format code with **Prettier** and detect issues early using **ESLint**

- Manage Git commits and branches visually
- Use the integrated terminal for executing build scripts and testing APIs
- Work with multiple workspaces when juggling AI Videos and AI Discover

2. Postman

- Test API responses for text-to-speech, summarization, and PDF parsing
- Automate request flows for multi-step processes like login → upload → summarize
- Set environment variables to switch between local, staging, and production endpoints

3. Git & GitHub

- Branching strategy (feature/bugfix/staging)
- Pull Request reviews
- Commit message conventions (feat/fix/docs/refactor)
- Code review comments and feedback loops

4. Azure DevOps / Vercel CI Pipelines

- **Vercel** handled frontend deployment. Every new commit triggered a preview deployment URL, making it easier to test UI updates and verify feedback before going live.
- **Azure DevOps** was used to manage server-side workflows. It allowed scheduled builds, environment variable configurations, and auto-deployment to App Services or Azure Functions.

5. Online APIs and External Tools

- AssemblyAI for transcription (experimental testing)
- Google Translate API for real-time translation in multi-lingual responses
- Gemini / OpenAI APIs for summarization and text understanding

4.4 Learning Outcomes

Undertaking a full-time internship at Tangible IT, Mexico as an AI-Driven Full Stack Developer was not only a significant academic milestone but also a personal transformation. Working on advanced AI-integrated platforms like AI Videos and AI Discover exposed me to real-world complexities, team dynamics, and large-scale software architecture. Each challenge and task translated into a concrete learning opportunity, shaping me into a more confident and industry-ready engineer.

1. Full-Stack Development Proficiency

One of the most crucial learning gains from this internship was the holistic understanding of full-stack development. I worked extensively on both frontend (React.js) and backend (Node.js, Express) aspects, enabling me to:

- Build responsive and dynamic user interfaces using modern JavaScript frameworks.
- Manage state effectively using context APIs and hooks.
- Create and consume REST APIs for various modules like summarization, translation, and document analysis.
- Understand middleware, routing, and request handling in backend systems.

2. Mastery of Cloud and Database Systems

Before the internship, my exposure to cloud platforms and NoSQL databases was limited to academic exercises. However, working with Azure services, CosmosDB, and Vercel deployments provided me with:

- Hands-on experience with serverless architecture and managed deployments.
- Practical understanding of how NoSQL data is structured, queried, and optimized.
- Skills in schema configuration and handling persistence logic for chat histories.

3. AI and NLP Integration in Real Systems

A major highlight was contributing to AI-based features:

- Embedding Google Gemini or OpenAI LLMs for transcript summarization.
- Using text-to-speech and translation APIs for multilingual accessibility.
- Integrating AI-powered flashcard and mind map generation.

4. Exposure to Agile Workflows and Team Dynamics

From sprint planning to code reviews and stand-up meetings, I participated in an agile work culture where collaboration and communication were key. I learned to:

- Break tasks into manageable sprints using Notion and GitHub Projects.
- Accept and implement feedback from peer code reviews.
- Communicate blockers early to keep the pipeline flowing.
- Log progress and updates regularly using documentation tools.

5. Documentation and Communication

Professional development isn’t complete without effective communication. Through this internship, I learned:

- How to write internal documentation for deployment processes, error handling, and platform features.
- How to report bugs and suggest enhancements during team discussions.
- How to document API schemas, usage examples, and setup guidelines for future developers.

Chapter 5

RESULTS AND IMPACT

The culmination of any internship is not merely the successful completion of assigned tasks but the transformative impact it has on the intern's technical proficiency, collaborative skills, and professional outlook. This chapter captures the tangible and intangible outcomes of my full-time internship at Tangible IT, Mexico, where I served as an AI-Driven Full Stack Developer under the guidance of Luis Angel Silva. Through my involvement in two major AI-powered platforms—AI Videos and AI Discover—I experienced a significant evolution in my development capabilities, problem-solving strategies, and understanding of enterprise-scale project execution.

The results were measured not just in terms of deployed features or successful code merges, but in terms of the knowledge gained, contributions made to the company's products, the value delivered to end-users, and the personal growth I underwent during the internship journey. The internship bridged the gap between academic foundations and real-world implementations, allowing me to engage in agile team dynamics, handle live deployments, and tackle technical bottlenecks head-on.

5.1 Technical Accomplishments

The most immediate impact of this internship was reflected in the technical milestones achieved across both projects. My contributions spanned frontend and backend development, cloud integration, and real-time UI feature enhancements, all of which made visible improvements in the applications' overall performance and usability.

Key Technical Outcomes:

- Full-Stack Ownership: I was entrusted with complete ownership of feature modules, including content-type switching logic, conversation history rendering, and UI state handling.
- Cross-Content Integration: Developed the ability to handle YouTube, PDF, and image inputs in a single workflow, enhancing the flexibility of the AI Videos platform.
- Persistent Conversation Logic: Enabled secure and stable conversation saving and retrieval through CosmosDB, ensuring seamless user experiences in AI Discover.
- Optimized API Utilization: Reduced unnecessary API calls and re-renders by refining backend workflows, which led to faster content processing and response.

These accomplishments solidified my understanding of component-driven architecture, state management, error handling, and the performance optimization required for enterprise-scale applications.

5.2 Product Impact and Value Addition

Beyond personal achievements, my work directly contributed to the enhancement of both platforms. These applications were not conceptual demos; they were actively deployed tools designed to assist learners, educators, and professionals with powerful AI-based utilities.

User-Centric Enhancements:

- Improved User Navigation: Refined navigation logic between input types and tabs prevented user confusion and ensured a smoother experience.
- Advanced Search Experience: The integration of real-time search, accurate citations, and fact-checked answers in AI Discover made the chat assistant much more reliable for research and information retrieval.
- Educational Enablement: AI Videos became a comprehensive platform for educational content transformation, enabling flashcards, summaries, and mind maps to be generated from simple multimedia inputs.

My work helped convert these ambitious goals into tangible interfaces and features, adding real business and functional value to the tools offered by Tangible IT.

5.3 Personal and Professional Growth

This internship marked a significant phase of personal evolution. It challenged my comfort zone, taught me how to collaborate with experienced engineers in a global setting, and demanded quick adaptation to new technologies and agile environments.

Growth Areas:

- Professional Discipline: Working six days a week under tight schedules enhanced my time management, accountability, and ability to meet deadlines without compromising quality.
- Team Collaboration: Frequent code reviews, planning discussions, and feedback cycles helped me understand the dynamics of team-based software development.
- Tech Confidence: I became confident in handling production-level systems, debugging live issues, and proposing changes to improve existing functionality.

This professional maturity was a direct result of being continuously exposed to the high standards and expectations of Tangible IT, which emphasized quality, innovation, and collaborative problem-solving.

5.4 Real-World Exposure and Learning Outcomes

One of the most valuable takeaways from this internship was the level of exposure to enterprise workflows and the real-world implementation of software systems. From requirement analysis to deployment, every stage of the software lifecycle was touched upon during my time at Tangible IT.

Major Learning Outcomes:

- Agile and Sprint Practices: I actively participated in sprint planning, task assignments, and feature prioritization cycles that mirrored real industrial practices.
- Problem-Solving with Ownership: Rather than being assigned micro-tasks, I was expected to take full ownership of feature development, bug resolution, and performance tuning.
- Documentation and Knowledge Transfer**:** Writing developer-facing documentation for both applications helped me master the art of clear technical communication.

These experiences positioned me to transition smoothly from an academic setting into the corporate world, equipped with not only technical skills but also industry-relevant practices and mindsets.

5.5 Broader Impact on Career Aspirations

This internship served as a career-defining experience that provided clarity and direction to my professional goals. By working on AI-powered content systems and conversational agents, I developed a strong inclination towards full-stack development and intelligent systems.

Career Impacts:

- Specialization Clarity: I identified my interest in building scalable, intelligent software systems, leading me to pursue deeper learning in AI-driven applications and enterprise architecture.
- Resume and Portfolio Enhancement: The projects I worked on added credible and impressive entries to my resume, supported by real features I helped develop and deploy.
- Professional Identity: I evolved from a student to a confident developer ready to contribute meaningfully in a professional, high-stakes environment.

CONCLUSION

The internship at Tangible IT, Mexico marked a significant milestone in my professional journey, transforming theoretical knowledge into impactful real-world applications. As a full-time AI-driven Full Stack Developer, I had the opportunity to contribute meaningfully to two major projects—AI Videos and AI Discover—that demonstrated the power of artificial intelligence in content processing and conversational intelligence. Through rigorous development tasks involving frontend and backend engineering, cloud deployment, API optimization, and database configuration, I deepened my expertise in technologies like React.js, Node.js, CosmosDB, and Azure. The collaborative environment, guided by industry professionals like Luis Angel Silva, instilled in me a strong understanding of agile workflows, scalable architecture, and efficient problem-solving.

This internship was not only a platform to build technical skills but also a catalyst for professional growth. Overcoming challenges such as schema issues, user session handling, and multi-input processing taught me resilience and adaptability. I also enhanced my communication, documentation, and team coordination abilities—essential traits for any software engineer. Ultimately, this experience has empowered me with the confidence and competence to work on enterprise-grade applications and positioned me to make meaningful contributions to the ever-evolving tech industry. It stands as a cornerstone of my career, paving the way for future innovation and leadership in the field of software development.

REFERNCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016. ISBN: 9780262035613.

[2] F. Chollet, Deep Learning with Python, 2nd ed., Manning Publications, 2021. ISBN: 9781617296864.

[3] A. Patterson, Foundations of Artificial Intelligence in Python, O'Reilly Media, 2020. ISBN: 9781492078197.

[4] Microsoft Azure Documentation – CosmosDB, https://learn.microsoft.com/en-us/azure/cosmos-db/

[5] J. Duckett, JavaScript and jQuery: Interactive Front-End Web Development, Wiley, 2014. ISBN: 9781118531648.

[6] E. Freeman, E. Robson, B. Bates, and K. Sierra, Head First Design Patterns, 2nd ed., O'Reilly Media, 2021. ISBN: 9781492078005.

[7] AssemblyAI API Documentation – Speech-to-Text & NLP APIs, https://www.assemblyai.com/docs/