PROJECT SYNOPSIS

On

AiTutor: AI POWERED LEARNING TUTORIAL FINDER WEBSITE

Submitted for partial fulfilment of requirement for the

award of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING (2026)

Submitted By

Amarjeet (2301220109004)
Aun Raza Khan (2301220109007)
Mantasha Islam (2301220109013)

Under the Guidance of

Er. Ratan Rajan Srivastava

SHRI RAMSWAROOP MEMORIAL COLLEGE OF ENGINEERING AND MANAGEMENT LUCKNOW,
Affiliated to,

Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW, LUCKNOW, UTTAR PRADESH

TABLE OF CONTENTS

1.INTRODUCTION

Education in the 21st century has undergone a radical transformation, largely influenced by the advent of digital platforms and intelligent technologies. The shift from traditional classroom-based learning to online and hybrid models has created immense opportunities for learners across the globe. Today, millions of students rely on platforms such as **YouTube, Coursera, edX, Udemy, and educational blogs** to gain access to knowledge in diverse technical and non-technical domains. While this transformation has democratized education, it has also introduced a significant challenge: **the problem of information overload.** Students often struggle to identify the most relevant tutorials and courses tailored to their unique learning requirements.

Artificial Intelligence (AI) and Natural Language Processing (NLP) offer effective solutions to this challenge. These technologies enable the development of **intelligent educational platforms** capable of analyzing user intent, interpreting natural language queries, and retrieving the most relevant educational content. AI-powered systems not only filter out irrelevant data but also personalize the recommendations based on user needs. In recent years, the integration of **Large Language Models (LLMs)**, such as OpenAI's GPT, with frameworks like **LangChain**, **Web Scraping** has revolutionized how queries can be understood and processed for meaningful outcomes.

The rapid growth of online education platforms has shown the importance of personalized learning. For example, while Coursera and edX provide structured learning pathways, they lack fine-grained customization at the individual learner level. YouTube, on the other hand, hosts millions of tutorials, but searching for quality and relevant content becomes cumbersome without an intelligent filtering mechanism. Hence, there is a strong need for an **AI-powered learning tutorial finder website** that bridges this gap by intelligently scraping course data, understanding student queries, and providing the most relevant tutorials in real-time.

In the field of **personalized education**, AI has been recognized as a transformative force. AI has the potential to redefine the roles of both teachers and learners by creating adaptive and tailored learning environments. With the integration of NLP, educational systems can better understand student intent, leading to more accurate recommendations.

Furthermore, the rise of **web scraping techniques** has enabled the extraction of vast amounts of data from multiple platforms. By combining web scraping with AI models, it becomes possible to build a robust system capable of analyzing, filtering, and presenting educational content efficiently.

The use of **vector similarity search** and **semantic embeddings** ensures that the system is not merely keyword-based but can also capture the context of user queries. This provides a level of personalization that is missing in most current platforms.

2.LITERATURE REVIEW

2.1 Overview: The rapid expansion of online educational resources has driven research into methods for organizing, filtering, and recommending learning content. Traditional recommendation systems used by massive open online course (MOOC) platforms rely mainly on collaborative filtering, content-based filtering, and popularity metrics. While these approaches perform well for large-scale, user-behavior-driven recommendations, they often fail to capture fine-grained user intent expressed in natural language and lack the contextual understanding needed for personalized tutorial discovery.

2.2 Recommender Systems in Education: Classical recommender systems in educational settings typically use user profiles, course metadata, and interaction history to suggest courses. Content-based filtering leverages course descriptions and tags, and collaborative filtering exploits similarities in user behavior. These systems are effective when user interaction data is abundant, but they struggle for new users or for ad-hoc queries (the cold-start problem) and cannot interpret detailed natural-language requests (e.g., "beginner Python tutorial with data-analysis projects").

2.3 Web Scraping and Content Aggregation: Several works emphasize the importance of automated content aggregation from multiple sources to build a richer knowledge base. Web scraping tools and APIs make it possible to gather diverse tutorial metadata (titles, descriptions, transcripts, ratings). However, scraped data is often unstructured, noisy, and heterogeneous across platforms; effective preprocessing, deduplication, and normalization are therefore essential steps before any recommender/semantic system can use the data reliably.

2.4 Semantic Retrieval and Embedding-Based Search: Recent advances in representation learning introduced vector embeddings and semantic similarity search as more robust alternatives to keyword matching. Embedding-based retrieval captures semantic relationships between queries and documents, enabling the system to return contextually relevant resources even when lexical overlap is low. Such vector search methods (using cosine similarity or approximate nearest neighbors) have become the backbone of modern retrieval-augmented systems.

2.5 Large Language Models (LLMs) and Retrieval-Augmented Generation: The advent of large language models (LLMs) has significantly improved query

understanding and summary generation. Combining LLMs with retrieval (RAG—retrieval-augmented generation) allows systems to both find relevant content and synthesize concise, user-friendly responses. Frameworks that orchestrate chunking, embedding generation, and LLM calls (for example, commonly used developer frameworks) enable scalable pipelines that respect token limits while maintaining high relevance and natural phrasing in the output.

2.6 Gaps in Existing Systems: Although MOOC platforms, search engines, and video portals provide massive amounts of content, several limitations persist: Most platforms rely on metadata and popularity rather than deep semantic matching, which reduces precision for detailed or niche queries. Few systems combine continuous web scraping, semantic embeddings, and LLM-based query interpretation in a unified, user-facing application targeting tutorial discovery. There is limited tooling focused on the educational value dimension (skill level, project-based learning fit, and learning outcomes) rather than only topic relevance. Practical constraints — API limits, dynamic web content, token restrictions in LLMs — require careful engineering (chunking, embedding stores, caching) that many prototype systems do not fully address.

2.7 How This Project Builds on Prior Work: The proposed AI-Powered Learning Tutorial Finder integrates several strands of prior research and engineering practice, It uses automated web scraping to aggregate a wide and up-to-date corpus of tutorials and metadata. It applies embedding-based semantic retrieval to match user intent with content beyond surface keywords. It leverages LLMs for query understanding, result summarization, and ranking, mitigating cold-start issues and improving usability for natural-language queries. By incorporating text chunking, embedding stores, and a performance analysis pipeline, the design addresses practical limits (token/window sizes, API usage) that often hamper real deployments.

3.PROBLEM DEFINITION

In today's digital era, a vast amount of educational content is available across multiple online platforms such as YouTube, Coursera, edX, Udemy, blogs, and technical forums. While the availability of such content has democratized access to education, it has simultaneously created the problem of **information overload**. Learners are often overwhelmed by the sheer volume of tutorials, making it difficult to select the most relevant and high-quality resources.

Another issue lies in **lack of personalization**. Current recommendation systems on educational platforms are generally category-based or rating-based, offering suggestions that may not align with the unique learning goals of individual students. This one-size-fits-all approach reduces the effectiveness of self-learning.

The rapid growth of online learning platforms has created an overwhelming abundance of tutorials, courses, and educational resources. However, learners often face challenges in discovering the most relevant and high-quality tutorials suited to their skill level, learning style, and goals.

Traditional search engines and MOOC platforms generally rely on keyword-based matching or popularity-based rankings, which fail to capture the context and intent behind user queries. As a result.

Personalized recommendation and semantic understanding of queries are largely absent. This creates a pressing need for an intelligent, automated system that can collect tutorials from multiple sources, process them using NLP and AI, and recommend the most suitable tutorials based on user intent and context.

4. PROJECT OBJECTIVE

- Create a web application that allows students to input their learning queries in natural language and receive precise, high-quality tutorial suggestions.
- Ensure the recommendations are tailored to individual student needs based on the specificity of their queries, such as skill level.
- Develop a platform that can be easily accessed by students worldwide, helping them overcome the challenge of information overload and focus on meaningful learning.
- Build a frontend that displays the recommended tutorials clearly, with options to filter or refine results.
- Implement web scraping and APIs to continuously gather tutorials from trusted educational sources such as YouTube, MOOCs, and blogs.
- Use NLP techniques and LLMs (e.g., GPT with embeddings) to interpret natural-language user queries.
- Clean, normalize, and categorize tutorials into structured formats with metadata (topic, difficulty level, duration, etc.).
- Develop a retrieval-augmented recommendation system using embeddings and vector similarity for precise content matching.
- Build a web-based interface where users can search tutorials in natural language and receive relevant recommendations with summaries.
- Evaluate system accuracy, relevance, and response time using real datasets and user feedback.
- Deploy the solution on cloud infrastructure ensuring cost-effectiveness, scalability, and continuous updates.

5. PROPOSED METHODOLOGY

The development of the *AI-Powered Learning Tutorial Finder Website* follows systematic **Software Engineering principles** to ensure scalability, maintainability, and efficiency. The methodology can be explained using **Software Development Life Cycle (SDLC)** phases:

5.1 Requirement Analysis:

- Gather user requirements such as the ability to input natural language queries and receive relevant tutorials.
- Identify functional requirements (web scraping, NLP processing, recommendation engine, web interface).
- Define non-functional requirements (response time, accuracy, scalability, usability).

5.2 System Design

- **High-Level Design (HLD):**
    - Architectural components: User Interface, Backend API, Web Scraper, NLP Engine, Database, Recommendation Engine
    - Data flow representation using **DFD (0-level and Level-1)**.
- **Low-Level Design (LLD):**
    - Detailed database schema for tutorial storage.
    - Algorithms for similarity search (cosine similarity on embeddings).
    - Workflow integration between LangChain, GPT, and vector storage.

5.3 Implementation

- **Frontend Development:** Create a user-friendly web interface for query input and result display.
- **Backend Development:** Develop APIs using Flask/Django to connect modules.
- **Web Scraping:** Implement BeautifulSoup and Playwright scripts for tutorial collection.
- **NLP Module:** Integrate LangChain + OpenAI GPT for query understanding and response generation.

- **Recommendation Engine:** Implement embedding generation and similarity search for ranking tutorials.

5.4 Testing

- **Unit Testing:** Test individual modules like scraping, NLP, and recommendation engine.
- **Integration Testing:** Verify interactions between frontend, backend, and databases.
- **System Testing:** Ensure the system works as a whole according to requirements.
- **Performance Testing:** Measure accuracy, response time, and scalability.

5.5 Deployment

- Deploy the application on cloud infrastructure (AWS/Azure/Google Cloud).
- Configure domain, hosting, and SSL for secure access.
- Ensure continuous integration and regular updates via automated scraping.

5.6 Maintenance & Evaluation

- Collect user feedback for improving accuracy.
- Update scraping sources and retrain embeddings as required.
- Fix bugs, enhance features, and improve multi-language support in future versions.

BLOCKDIAGRAM OF PROPOSE SYSTEM

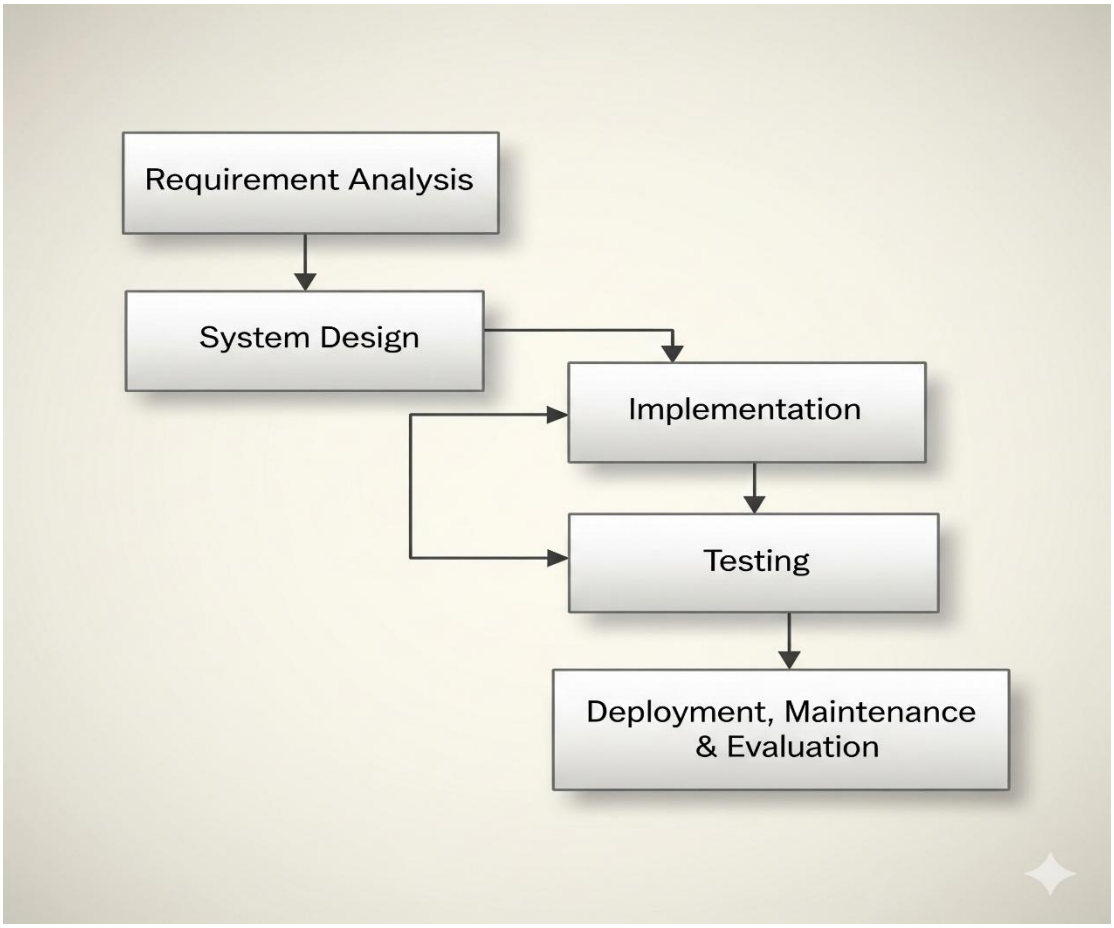


Figure 2 (SDLC Workflow Diagram on My project Deployment)

6. ALGORITHM & TECHNOLOGY USED

The functioning of the AI-Powered Learning Tutorial Finder Website is driven by multiple algorithms that ensure effective content retrieval, processing, and recommendation.

6.1 Algorithm Used:

Web Scraping Algorithm

- **Input:** Target tutorial source URLs (MOOCs, YouTube, blogs).
- **Process:**
    1. Send HTTP requests or automate browser using Playwright.
    2. Parse HTML/JSON responses using BeautifulSoup.
    3. Extract metadata (title, description, tags, ratings).
    4. Store structured tutorials in the database.
- **Output:** A continuously updated tutorial knowledge base.

Natural Language Processing (NLP)

- **Input:** User query (e.g., *"beginner Python tutorial with projects"*).
- **Process:**
    1. Tokenization and semantic parsing of query.
    2. Generate embeddings using LLM.
    3. Compare embeddings against tutorial database.
    4. Rank tutorials based on similarity scores.
- **Output:** Contextually relevant tutorials ranked for the user.

Embedding & Similarity Search Algorithm

- Generate vector embeddings for both tutorials and queries.
- Store embeddings in a **vector database** for efficient retrieval.
- Perform **cosine similarity** to identify top matches.
- Retrieve top-N tutorials for recommendation.

6.2 Technology Used:

The development of the **AI-Powered Learning Tutorial Finder Website** integrates modern technologies from **web scraping, natural language processing, large language models, and web deployment** domains. These technologies ensure automated data collection, semantic understanding of queries, and user-friendly tutorial recommendations.

Web Scraping Tools

- **BeautifulSoup (Python Library):**
    - Used to parse and extract structured content (titles, descriptions, tags) from static web pages.
    - Enables efficient navigation of HTML and XML documents.
- **Playwright (Browser Automation Tool):**
    - Automates interactions with dynamic websites that rely on JavaScript.
    - Ensures that tutorials hosted on modern web platforms are also scraped accurately.

Natural Language Processing (NLP) Framework

- **LangChain:**
    - Manages interaction between the GPT model and backend databases.
    - Handles **text chunking** for processing long tutorial transcripts.
    - Generates embeddings for both queries and tutorial content.
    - Enables **similarity-based retrieval** of content.

Deployment Technologies

- **Flask/Django:** Backend frameworks used to handle API requests, integrate NLP modules, and serve results to the frontend.
- **Cloud Hosting (AWS/Azure/Google Cloud):** Ensures scalability, accessibility, and reliable performance for global users.
- **Domain & SSL:** Provides a secure and user-friendly website interface.

7.HARDWARE & SOFTWARE REQUIREMENT

For the successful development and deployment of the **AI-Powered Learning Tutorial Finder Website,** both software and hardware resources are required. These requirements ensure smooth execution, efficient processing, and user accessibility.

7.1 Software Requirements

1. Programming Language: Python 3.x

- Chosen for its extensive libraries in AI, NLP, and web development.

2. Libraries and Frameworks:

- **LangChain** – For managing LLM workflows and query processing.
- **OpenAI GPT API** – For natural language understanding and tutorial recommendation.
- **BeautifulSoup & Playwright** – For web scraping and dynamic content extraction.
- **Requests** – For handling HTTP requests while scraping data.
- **Flask/Django** – For backend web development and API management.

3. Development Environment:

- Jupyter Notebook / VS Code / PyCharm.

4. Operating System:

- Windows 10/11 or Linux-based OS.

7.2 Hardware Requirements

- **Processor:** Intel i5 (8^{th} Gen or above) / AMD equivalent.
- **Ram:** Minimum 8 GB (Recommended: 16 GB for efficient NLP processing).
- **Storage:** At least 250 GB / 256 GB SSD for faster data handling.
- **Internet Connection:** Stable broadband connection for API access and scraping.

8. MODULE DESCRIPTION

The proposed system consists of multiple modules, each performing a specific task. Together, these modules ensure smooth functioning of the AI-powered tutorial recommendation platform. Below is a detailed description of each module including input, output, and functioning.

8.1 Query Input Module

- **Proposed Input:** A natural language query entered by the learner (e.g "Beginner Python tutorial with data analysis examples")
- **Functioning:**
    - The user interacts with a simple web interface.
    - The input is captured through a text field and passed to the backend system.
    - The query is then preprocessed (removing unnecessary characters, normalizing text) before being forwarded to the NLP module.
- **Proposed Output:** A clean, structured query that can be processed further by the NLP and similarity search modules.

8.2Web Scraping Module

- **Proposed Input:** Educational content from online platforms such as YouTube, Coursera, edX, and blogs.
- **Functioning:**
    - This module uses **BeautifulSoup** for static pages and **Playwright** for dynamic pages.
    - It extracts tutorial titles, descriptions, durations, tags, and direct links.
    - The processed data is stored in a structured database for later use.
- **Proposed Output:** A database of tutorials containing relevant metadata (title, description, link, category, skill level).

8.3 NLP Processing Module

- **Proposed Input:** Preprocessed user query with tutorial content extracted by the web scraping module.

- **Functioning:**
    - Uses **LangChain** to manage query workflow.
    - Both user queries and tutorial content are converted into **vector embeddings**.
    - **Text chunking** is applied to break large tutorial descriptions into smaller, manageable pieces.
- **Proposed Output:** Context-aware vector representations of both queries and tutorial content, ready for similarity matching.

8.4 Recommendation Engine (Similarity Search Module)

- **Proposed Input:** Embeddings of user queries of tutorials from the database.
- **Functioning:**
    - Performs **cosine similarity search** between query embeddings and tutorial embeddings.
    - Identifies the closest matches based on semantic meaning.
    - GPT refines the results, ranks tutorials, and prepares a summarized list of the most suitable ones.
- **Proposed Output:** A ranked list of the most relevant tutorials, each containing title, link, and a short description.

8.5 Frontend & Deployment Module

- **Proposed Input:** Tutorial recommendations generated by the Recommendation Engine.
- **Functioning:**
    - The result are displayed through a **user-friendly web interface**.
    - Filter may be provided to refine results (e.g., beginner/intermediate/advanced).
    - The system also collects user feedback for performance evaluation.
- **Proposed Output:** A clear and interactive webpage displaying AI- recommended tutorials, accessible in real-time.

8.6 Performance Analysis Module

- **Proposed Input:** Generated tutorial recommendations and user feedback and system log.
- **Functioning:**
    - Evaluates system accuracy by checking how many recommendations match the user's learning goals.
    - Calculates metrics such as **Precision, Recall, and Response Time**.
    - Collects user satisfaction ratings to improve system performance.
- **Proposed Output:** A performance report that highlights system efficiency, accuracy, and areas of improvement.

9.Diagram

9.1 0-Level DFD:

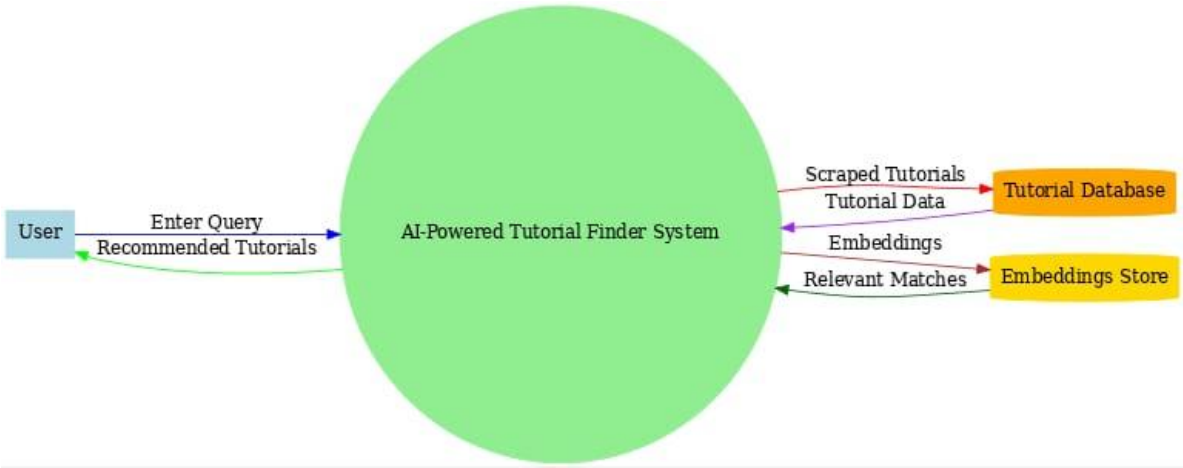


Figure 3 (DFD detail on Tutorial Finder Website)

9.2 Use Case Diagram

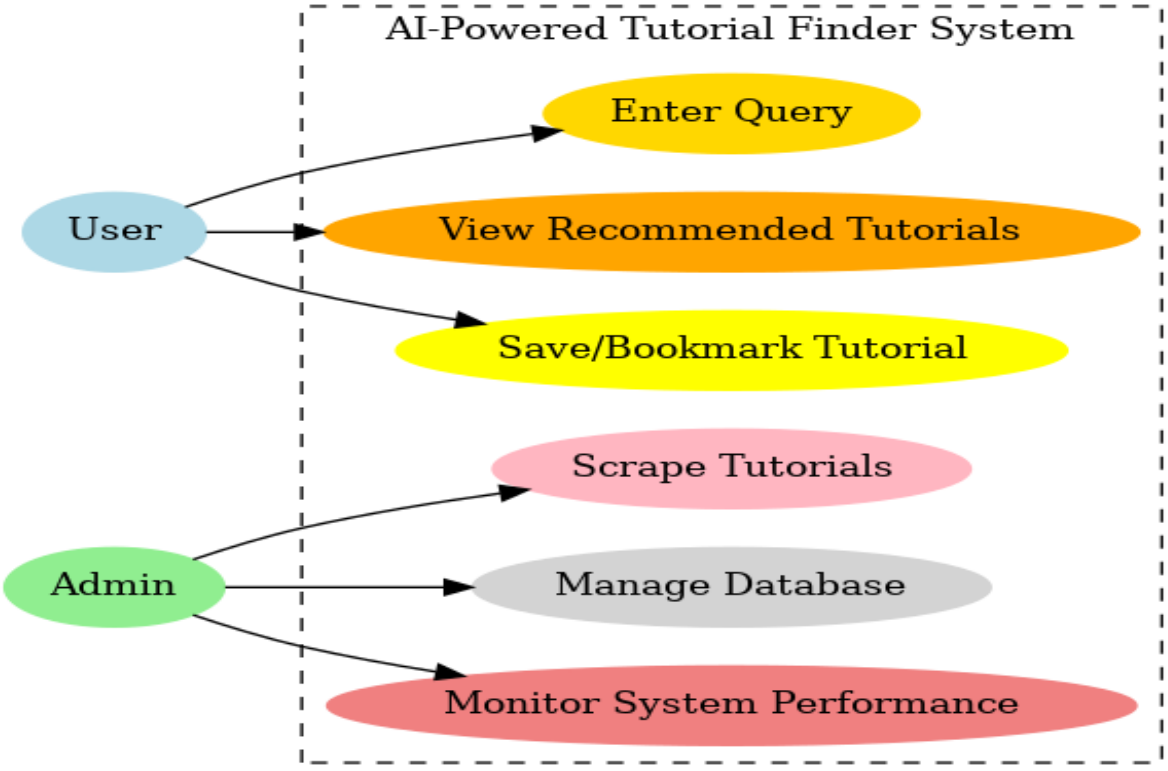


Figure 4 (Working hole project scenrios)

10. APPLICATION/ADVANTAGES/LIMITATIONS AND PROPOSED COST

10.1 Application:

1. Personalized Education Platforms – Can be integrated into online learning portals like edX, Coursera, or university website provide customized tutorial.

2. Skill Development and Training – Helps learners in technical fields such as programming, AI, and data science by recommending the most relevant resources.

3. Academic Research Assistance – Students and researchers can quickly locate relevant tutorials and resources for projects or thesis work.

4. Corporate Training Systems – Organizations can use the system to recommend specific tutorials for employee skill upgradation.

5. E-Libraries and Knowledge Portals – Enhances digital libraries by adding an intelligent tutorial discovery features.

10.2 Advantages:

1. Time-Saving – Eliminates the need to manually browse through thousands of tutorials.

2. **Context-Aware Recommendations –** Unlike keyword-based search, it provides semantic and meaningful matches.

3. Scalability- Can be extended to multiple domains and educational sources.

4. Automation- Database of tutorials is Simplifies learning by offering a clean and intuitive interface.automatically updated using scraping, ensuring the latest resources are available.

10.3 Limitations:

1.Dependency on APIs and Web Sources: If data source (e.g, YouTube API restrict access, scraping may face limitations.

2. Accuracy Variability: Recommendations rely on the quality of training data and embeddings; sometimes irrelevant tutorials may appear.

3.Computational Cost: NLP and similarity search require significant computational power for large-scale use.

4.Language Limitation: Current implementation focuses mainly on English content; multi-language support may need future improvements.

10.4 Proposed Cost Estimate:

The estimated cost for implementing AI Powered learning Tutorial Finder web application is as follow.

Component	Estimated Cost (INR)	Remarks
Software Tools (Python, BeautigulSoup, Playwright, LanChain, etc.)	₹0	Open-source libraries for scraping, automation, and NLP integration.
OpenAI GPT API (for NLP & query processing)	₹3,000–₹5,000 / month	Cost varies based on number of API calls and query load.
Hardware (Mid-level PC/Laptop: i5/i7, 8–16 GB RAM, 250–512 GB SSD)	₹50,000–₹70,000 (one-time)	Required for local development, training, and testing.
GPU Support (optional)	₹25,000–₹40,000 (optional)	For faster model execution and AI training acceleration.
Domain & SSL Certificate	₹800–₹1,200 annually	For secure domain registration and HTTPS encryption.
Maintenance (Broadband Internet Connection)	₹800–₹1,200 / month	Needed for development, deployment, and real-time data updates.

Total Estimate Cost (Prototype):

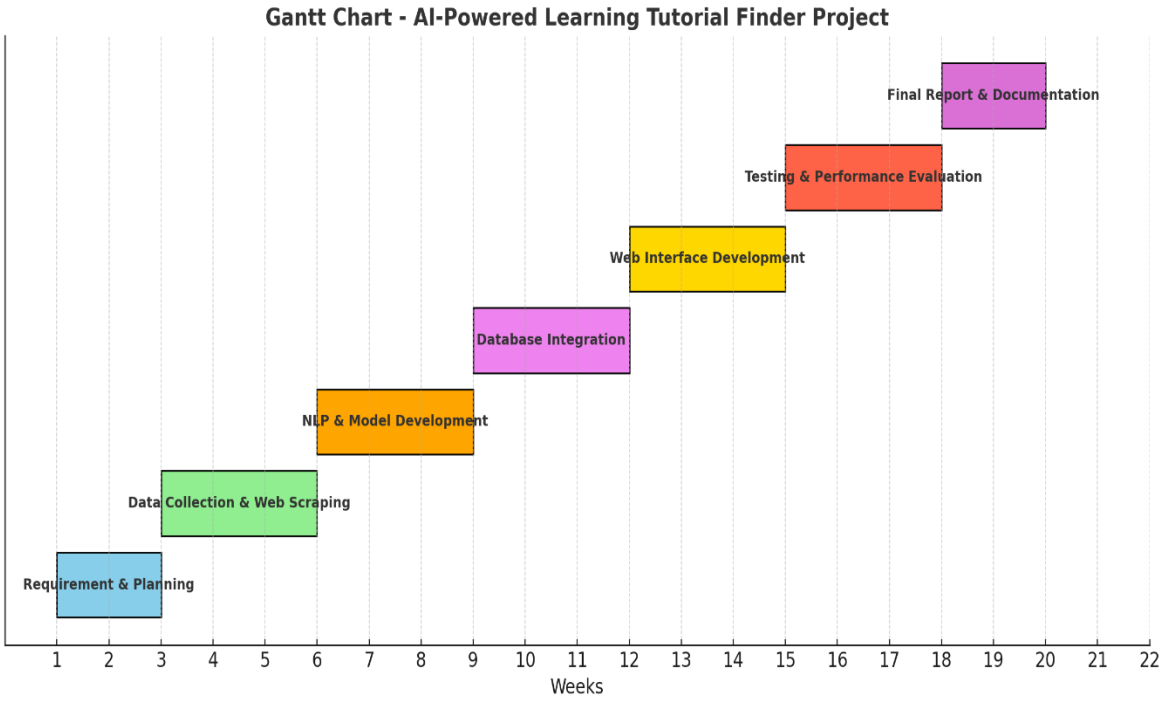
₹55,000–₹85,000 (approx.)

(Scales higher for enterprise use depending on data volume and recommendation and learning services)

11. GANTT CHART

11.1Time Performance Management on Project:

The Gantt Chart represents the project timeline and task scheduling for the AI-Powered Learning Tutorial Finder Website. It shows the sequential flow of activities from planning, data collection, and model development to deployment, testing, and final documentation.



Gantt chart:(Project Performance Timeline)

12. REFERENCE

1) Yunanto, A. A. (2017). Kecerdasan Buatan Pada Game Edukasi Untuk Pembelajaran Bahasa Inggris Berbasis Pendekatan Heuristik Similaritas [Unpublished Thesis, Institut Teknologi Sepuluh Nopember]. https://repository.its.ac.id/2072/
2) Zhang, Z. (2021). The Impact of Digital Technologies on Entrepreneurship Education. 448–452. https://doi.org/10.2991/assehr.k.210407.088
3) YouTube API Documentation – Google Developers. https://developers.google.com/youtube
4) Russell, M. A. (2013). Mining the Social Web. O'Reilly Media.
5) T. Baker and L. Smith, ''Educ-AI-tion rebooted? Exploring the future of artificial intelligence in schools and colleges,'' Nesta Found.,London, U.K., Tech.Rep.,2019.[Online].Available: https://media.nesta.org.uk/documents/Future_of_AI_and_education
6) X. Wang, M. Younas, Y. Jiang, M. Imran, and N. Almusharraf,''Transforming education through blockchain: A systematic review of applications, projects, and challenges,'' IEEE Access, vol. 13,pp. 13264–13284, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10804766
7) M. Afzaal, X. Shanshan, D. Yan, and M. Younas, ''Mapping artificial intelligence integration in education: A decade of innovation and impact (2013–2023)—A bibliometric analysis,'' IEEE Access, vol. 12,pp. 113275–113299, 2024, doi: 10.1109/ACCESS.2024.3443313
8) T. Wang, S. Li, X. Huang, Z. Pan, and S. P. Lajoie, ''Examining students'cognitive load in the context of self-regulated learning with an intelligent tutoring system,'' Educ. Inf. Technol., vol. 28, no. 5, pp. 5697–5715,May 2023, doi: 10.1007/s10639-022-11357-1.
9) T. K. F. Chiu, Q. Xia, X. Zhou, C. S. Chai, and M. Cheng,''Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education,'' Comput. Education: Artif. Intell., vol. 4, Jan. 2023, Art. no. 100118, doi:10.1016/j.caeai.2022.100118.C. J. Khilare, S. N. Pawar, D. D.
10) Namdas, V. P. Gaikwad, "Rooftop Rainwater Harvesting" Dahivadi College Campus, Satara SWRDM (2012).