Script2Scene A Natural Language Interface for Educational Animation Generation

Md. Abrar Khan[1], Mandalozu Pranav[2], Dr. K. Rajitha[3], Dr. V. Subbaramaiah[4]
[1,2]*Student, Mahatma Gandhi Institute of Technology*
[3,4]*Assistant Professor, Mahatma Gandhi Institute of Technology*

***Abstract*—In recent years, artificial intelligence has significantly advanced creative and educational technologies, enabling new modes of content generation and user interaction. This project, titled "Script2Scene: A Natural Language Interface for Educational Animation Generation," seeks to simplify the traditionally complex process of animation creation by allowing users to generate high-quality educational videos from plain text prompts. Manim, a powerful open-source Python library, is widely used for mathematical and conceptual animations but requires substantial programming expertise, limiting its accessibility. Script2Scene overcomes this barrier by integrating a Large Language Model (LLM) to automatically generate Manim-compatible scripts from natural language input. Script2Scene has strong educational relevance, offering educators, students, and content creators an accessible way to produce visual explanations, lectures, and learning materials without technical expertise.**

***Index Terms*—Educational Animation, Large Language Model, Manim, Natural Language Interface.**

I. INTRODUCTION

[Font: Times New Roman, Size:10]
Bb Creating educational animations traditionally demands programming, design expertise, and significant time investment, making it difficult for many educators and creators to visually explain complex STEM and other domain concepts. This limits access to high-quality visual learning tools that can simplify abstract topics for students. By removing technical barriers, this approach democratizes animation creation and enhances the way knowledge is communicated visually.

To overcome this, a natural language interface enables users to generate educational animations simply by describing scenes in plain text. Powered by large language models (LLMs) for automatic code generation and Manim for rendering, the system produces MP4 videos without requiring any coding knowledge, making the animation creation process fast, accessible, and efficient. This innovation empowers educators, improves learner engagement, and bridges the gap between creativity and technology in education.

II. LITERATURE REVIEW

Recent research has extensively explored the use of large language models (LLMs) and generative AI for code synthesis, program understanding, and content generation. Early works such as [5] introduced models like CodeBERT that bridge natural and programming languages, enabling tasks like code search and summarization, while later advancements such as CodeT5 [3] improved contextual understanding through identifier-aware architectures. Large-scale models like Codex [1] and AlphaCode [2] demonstrated the capability of generating functional and competitive-level code, although limitations in logical correctness and reasoning persist. Additionally, models like InCoder [4] extended capabilities to code infilling and interactive editing, making them more practical for real-world development scenarios.

Overall, these studies [1]–[20] indicate rapid progress in leveraging LLMs for code and content generation. However, existing systems often lack seamless integration between natural language understanding, programmatic animation generation, and user-friendly interfaces. This gap motivates the proposed system,

which aims to combine LLM-based code generation with automated animation rendering to create an efficient and interactive platform for generating educational visualizations from textual descriptions.

III. METHODOLOGY

The proposed system, Script2Scene, is designed to convert natural language prompts into executable animation videos through an integrated pipeline of web interaction, AI-driven code generation, and secure rendering. The architecture combines a user-friendly frontend with a robust backend that processes inputs, generates animation scripts, and delivers final visual outputs in real time.

A. Frontend Interface and User Interaction

The frontend serves as the primary interaction layer between the user and the system. It is implemented as a browser-based web interface where users can input natural language prompts along with optional parameters. The interface is responsible for collecting user input, sending requests to the backend, and displaying the generated animation output. Once processing is complete, the final animation is rendered as an MP4 video and presented directly within the interface, ensuring a seamless user experience.

B. Prompt Processing and Code Generation

The backend begins by processing the user input through a prompt processor, which structures and refines the raw prompt to improve model understanding. This processed input is then passed to a large language model (Gemini), which generates Python-based animation scripts using the Manim framework. The generated code represents the logical and visual structure of the requested animation.

C. Code Validation and Secure Execution

To ensure reliability and safety, the generated code undergoes validation to check for syntax errors, logical inconsistencies, and potential security risks. Only verified code is allowed to proceed to execution. The system employs a secure sandbox environment to run the code in isolation, preventing any unintended impact on the host system and ensuring safe handling of dynamically generated scripts.

D. Rendering and Output Generation

The validated code is executed using the Manim animation engine within the sandboxed environment. Manim processes the script to generate animation frames, which are then compiled into a final MP4 video. This output is returned to the frontend and displayed to the user. The integration of automated code generation with a rendering engine enables the system to transform textual descriptions into structured, high-quality visual animations efficiently.

IV. SYSTEM DESIGN AND ARCHITECTURE

The Script2Scene system is designed as a modular and scalable pipeline that transforms natural language prompts into animated educational videos. It integrates user interaction, AI-driven code generation, validation, and rendering into a seamless workflow that ensures both usability and secure execution.

A. System Workflow

1. Input Processing: The system accepts user inputs in the form of natural language prompts and optional documents such as PDFs. These inputs are preprocessed and structured for further analysis.
2. Content Understanding: A combination of retrieval mechanisms and a large language model (LLM) is used to extract relevant information and generate a concise, animation-ready summary.
3. Script and Code Generation: The processed content is converted into a structured narration script, which is then transformed into executable Manim-based Python code using the LLM.
4. Validation and Execution: The generated code is validated for correctness and safety.
5. Rendering and Output: The Manim engine renders the animation, optionally synchronizes it with generated audio, and produces a final MP4 video for the user.

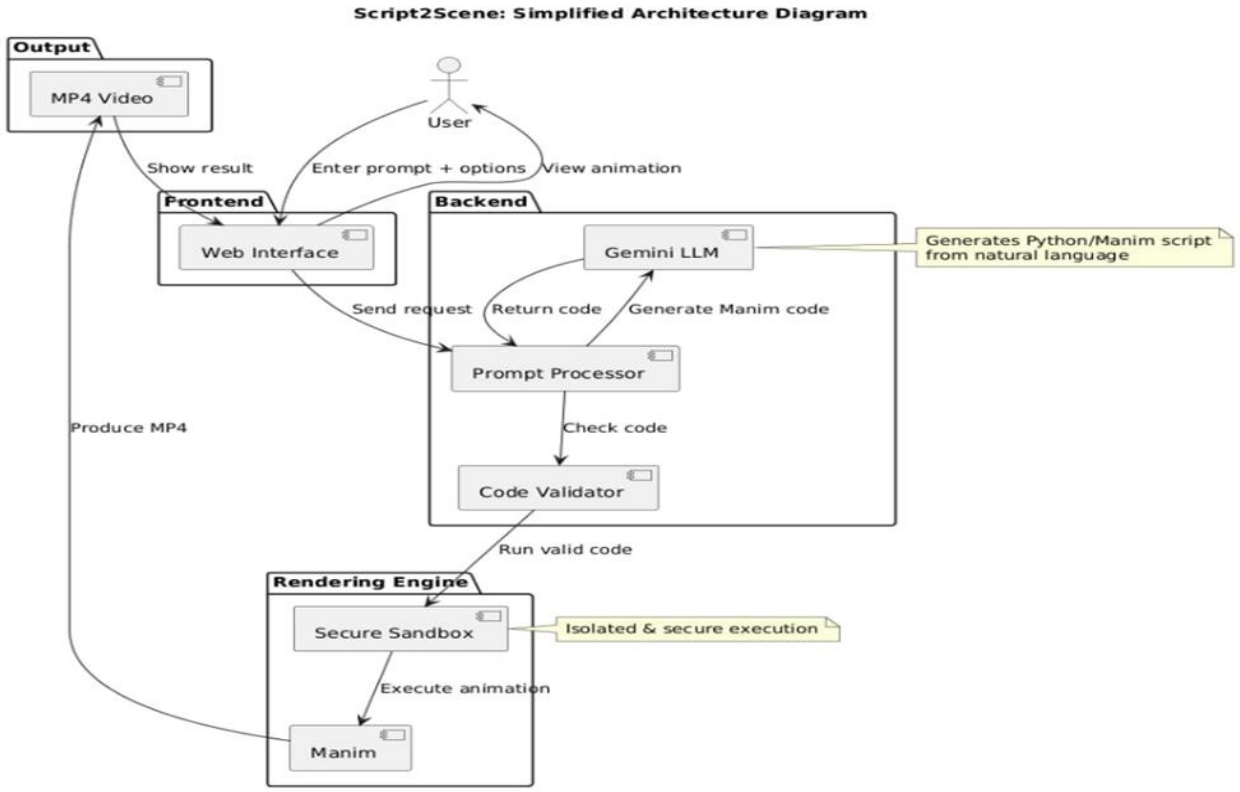

Fig. 1. Architecture of Script2Scene system showing data flow and model components

B. User Interface Components

The frontend interface enables users to interact with the system through a web-based platform. It allows users to:

• Enter prompts and upload supporting documents
• Customize animation parameters such as duration and style
• Preview generated animations directly in the browser
• Download the final MP4 video and view generated code

C. System Modules and Roles

• User: Initiates requests and interacts with the system
• Prompt Handler: Receives and validates user input
• LLM Engine: Generates scripts and animation code
• Code Validator: Ensures syntax correctness and runtime safety
• Rendering Engine: Executes code in a secure sandbox and generates video output
• Video Manager: Handles storage, preview, and download of generated videos

D. Design Goals

• Modularity: Each component operates independently, enabling easy updates and maintenance
• Security: Sandbox execution ensures safe handling of dynamically generated code
• Scalability: Supports future enhancements such as multilingual output and real-time processing
• Accessibility: Web-based interface allows usage without installation, ensuring ease of access

V. IMPLEMENTATION

The Script2Scene system is implemented as a full-stack web application that combines frontend interaction with backend AI processing and rendering capabilities.

A. Tools and Technologies

• Programming Language: Python
• Frontend: HTML, CSS, JavaScript (or React-based UI)
• Backend Framework: Flask / FastAPI
• Libraries
o AI Integration: Gemini API

o Animation: Manim
o Data Handling: JSON, Python standard libraries

• Deployment & Execution: Docker-based secure sandbox environment

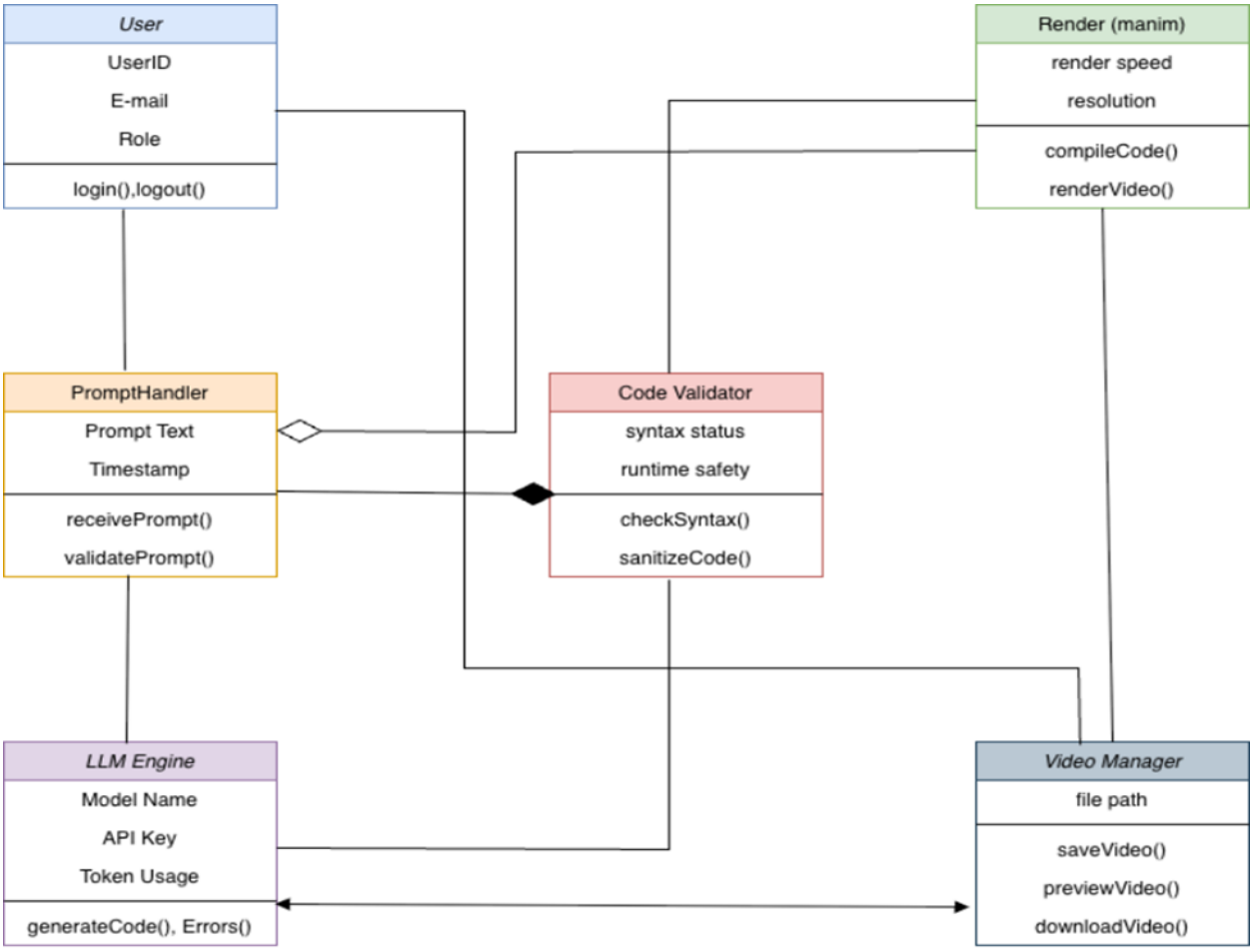

Fig. 2. Process flow diagram outlining flow of sequence from NLP input to Video generation

B. Code Generation and Rendering Pipeline
• The LLM (Gemini) generates Manim-compatible Python scripts from user prompts
• Generated code is validated for syntax and safety constraints
• Validated scripts are executed inside a sandboxed environment
• Manim renders animation frames and compiles them into MP4 format

C. Frontend Integration
The web interface facilitates real-time interaction by allowing users to input prompts and receive outputs dynamically. Key features include:
• Prompt input and customization controls
• Video playback with embedded player
• Optional display of generated source code
• Error handling and progress indicators

D. Execution Requirements
• System Specifications: Minimum 4GB RAM, multi-core processor
• Dependencies: Python environment with required libraries (Manim, API clients)
• Runtime Requirements: Stable internet connection for API calls and rendering support

This implementation ensures that Script2Scene operates efficiently as an end-to-end system, converting textual descriptions into high-quality animated visualizations through an automated and secure pipeline.

VI. RESULTS AND DISCUSSION

The performance of the Script2Scene system was evaluated based on functional correctness, system reliability, and overall user experience. Both quantitative observations and qualitative feedback were used to assess how effectively the system converts natural language prompts into animated visual outputs.

A. System Testing and Reliability

The system was tested using a combination of unit, integration, and end-to-end testing approaches. Individual modules such as prompt processing, code generation, validation, and rendering were tested independently to ensure correctness. Integration testing verified smooth communication between the frontend, backend, AI model, and rendering engine. End-to-end testing simulated real user scenarios, ensuring that the pipeline—from prompt input to final video output—operated without failures. Security testing confirmed that the sandbox environment safely handled dynamically generated code, while performance testing evaluated rendering times across varying prompt complexities.

B. Functional Output Evaluation

The system was validated using diverse educational prompts across domains such as mathematics, physics, and computer science. Each output was evaluated based on accuracy of concept representation, logical sequencing, and visual clarity. The generated animations successfully illustrated key concepts with proper labeling, smooth transitions, and adherence to user-defined parameters such as duration and resolution. The accompanying Manim code was also verified to be complete, readable, and reusable for further customization.

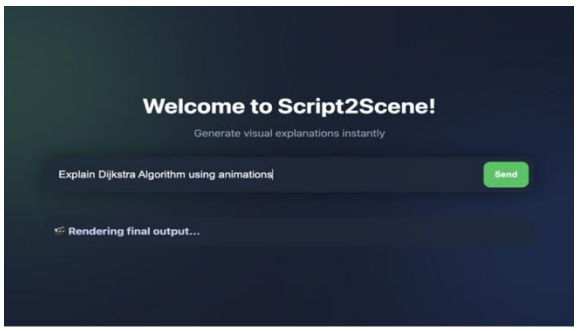

Fig. 3. Application page displaying prompt entering and video generation timeline

C. Visualization and Output Quality

Script2Scene produces high-quality MP4 animations that effectively translate textual descriptions into visual explanations. The outputs maintain consistency throughout the animation, ensuring that concepts are presented clearly from start to finish. The system supports multiple prompts and generates visually coherent results across different topics, demonstrating robustness and adaptability. The web interface further enhances usability by providing real-time playback, code visibility, and download options.

D. User Experience and Interface Features

The web-based interface allows users to easily input prompts, customize animation settings, and view results without technical expertise. Features such as loading indicators, error messages, and embedded video playback contribute to a smooth interaction experience. The system was tested across different scenarios, including invalid inputs and ambiguous prompts, and consistently provided informative feedback.

E. System Limitations

Despite strong performance, certain limitations were observed. Complex or highly abstract prompts may lead to less precise animations due to ambiguity in natural language interpretation. Rendering time increases with animation complexity, which may affect responsiveness under heavy load. Additionally, the system currently relies on predefined animation frameworks, which may limit flexibility for highly customized visualizations.

F. User Feedback

Feedback from users indicated that the system is intuitive and effective for learning purposes. Users appreciated the clarity of animations, ease of use, and the ability to generate visuals from simple text prompts. Suggestions for improvement included adding more customization options, supporting additional languages, and improving handling of highly complex concepts. Overall, the feedback validates the system's usefulness as an educational visualization tool.

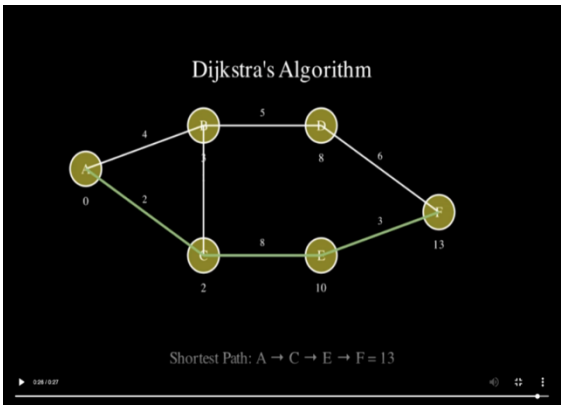

Fig. 4. Web Application home page displaying final result of video animation query

VII. CONCLUSION AND FUTURE WORK

A. Conclusion

This project presents Script2Scene, a web-based system that transforms natural language prompts into high-quality educational animations using Manim and a large language model. The system effectively bridges the gap between textual input and visual output by automating code generation, validation, and rendering within a unified pipeline.

The implementation demonstrates how AI-driven code synthesis can simplify animation creation, eliminating the need for programming expertise. The integration of a responsive frontend, intelligent backend, and secure sandboxed rendering environment ensures reliable performance and safe execution. Users can generate customized animations with control over parameters such as duration, style, and resolution, while also accessing the underlying code for further use.

Overall, Script2Scene highlights the potential of generative AI in enhancing learning by making complex concepts more accessible through visual explanations. It serves as an efficient tool for students, educators, and content creators, enabling rapid creation of educational animations from simple textual descriptions.

B. Future Work

Several improvements can further enhance the system's capabilities and scalability:

1. Longer and Structured Content Generation: Extending support for longer animations, such as full-length lectures, with improved scene management and continuity.
2. Performance Optimization: Reducing rendering time through GPU acceleration, parallel processing, and caching mechanisms to improve responsiveness.
3. Enhanced Automation: Incorporating features such as automatic voice narration, subtitle generation, and intelligent animation styling to create complete educational videos with minimal user effort.
4. Cloud-Based Deployment: Leveraging cloud infrastructure for scalable rendering and storage, enabling support for multiple concurrent users.
5. Multilingual and Accessibility Support: Expanding support for multiple languages and improving accessibility to reach a wider global audience.

By incorporating these enhancements, Script2Scene can evolve into a comprehensive and scalable platform for automated educational content generation, further advancing the role of AI in interactive learning systems.

REFERENCES

[1] M. Chen et al., "Evaluating Large Language Models Trained on Code," arXiv preprint arXiv:2107.03374, 2021.

[2] Y. Li et al., "Competition-level Code Generation with AlphaCode," Science, vol. 378, no. 6624, pp. 1092–1097, 2022.

[3] Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi, "CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code," arXiv preprint arXiv:2109.00859, 2021.

[4] D. Fried et al., "InCoder: A Generative Model for Code Infilling and Synthesis," arXiv preprint arXiv:2204.05999, 2022.

[5] Z. Feng et al., "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," Findings Assoc. Comput. Linguist.: EMNLP, pp. 1536–1547, 2020.

[6] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim, "A Survey on Large Language Models for Code Generation," arXiv preprint, 2023.

[7] Q. Zheng et al., "CodeGeeX: A Pre-Trained Model for Code Generation with Multilingual

Benchmarking on HumanEval-X," arXiv preprint arXiv:2303.17568, 2023.
[8] S. Lu et al., "CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation," arXiv preprint arXiv:2102.04664, 2021.
[9] S. Gulwani, O. Polozov, and R. Singh, "Program Synthesis," Found. Trends Program. Lang., vol. 4, no. 1–2, 2017.
[10] X. Wang et al., "Learning Abstractions for Program Synthesis," in Proc. POPL, 2017.
[11] U. Singer et al., "Make-A-Video: Text-to-Video Generation without Text-Video Data," arXiv preprint arXiv:2209.14792, 2022.
[12] J. Ho et al., "Imagen Video: High-Definition Video Generation with Diffusion Models," arXiv preprint arXiv:2210.02303, 2022.
[13] A. Ramesh et al., "Zero-Shot Text-to-Image Generation," arXiv preprint arXiv:2102.12092, 2021.
[14] B. Poole et al., "DreamFusion: Text-to-3D using 2D Diffusion," arXiv preprint arXiv:2209.14988, 2022.
[15] L. Huynh et al., "Detecting Code Vulnerabilities using LLMs," in Proc. IEEE/IFIP DSN, 2025.
[16] OpenAI, "HumanEval Dataset," 2021.
[17] V. Jain et al., "Sketch2Code: Generating Web Interfaces from Images," arXiv preprint arXiv:1910.08930, 2019.
[18] N. Bouali and V. Cavalli-Sforza, "A Review of Text-to-Animation Systems," IEEE Access, vol. 11, pp. 86071–86087, 2023.
[19] S. Bourgault et al., "Narrative Motion Blocks: Combining Direct Manipulation and Natural Language Interactions for Animation Creation," in Proc. DIS, 2025.
[20] K. Kousalya et al., "Text-to-Animation: Generating 3D Animation Using Textual Descriptions," IJRASET, 2025.