International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

CodeForesight: AI – Powered Learning and Coding Assistant

Om Awari[1], Serene D'Souza[2], Gaurav Kakad[3], Nikita Khuspe[4], Pratima Patil[5]

[1,2,3,4]Student, Department of Information Technology, Trinity Academy of Engineering, Pune
[5]Professor, Department of Information Technology, Trinity Academy of Engineering, Pune

Abstract
The increasing complexity of cybersecurity concepts poses a challenge for learners and professionals who seek clear, concise, and visual explanations. This paper presents CodeForesight, an AI-powered learning and coding assistant designed to generate visual representations and theoretical explanations of cybersecurity topics based on user prompts. Traditional learning methods often lack interactivity and adaptability to individual learning styles, especially when dealing with technical subjects like cybersecurity. Existing AI tools provide text-based assistance but rarely offer simultaneous visual support tailored to the query context. CodeForesight addresses this gap by delivering customized visual aids, helping users grasp abstract security mechanisms more intuitively. This dual-modality approach aims to improve cognitive retention and foster a deeper conceptual understanding. The system leverages a fine-tuned LLama model to provide responses with added capabilities for diagrammatic output and natural language processing. We describe the model training pipeline, data sources, architectural design, and interface implementation. Results indicate high user satisfaction and meaningful learning enhancement through visual outputs. This research contributes to the field of educational AI by integrating generative models into cybersecurity pedagogy.

Keywords: CodeForesight, Cybersecurity education, Generative AI, LLaMA model, Visual explanation, Educational assistant, Diagram generation, Natural language processing, Fine-tuned language model

1. Introduction

The continuous evolution of artificial intelligence (AI) and cybersecurity has introduced innovative avenues for transforming the learning experience in technical education. Despite these advancements, students often encounter significant difficulties in understanding abstract cybersecurity theories, conceptualizing network mechanisms, and correlating theoretical knowledge with practical implementation. These learning challenges highlight the need for a more intuitive and supportive educational platform tailored to complex technical domains. To address these issues, we introduce CodeForesight, an AI-powered educational assistant specifically developed to enhance students' comprehension of cybersecurity concepts through dynamic explanations and interactive visualizations. Traditional instructional approaches frequently struggle to bridge the gap between theoretical abstraction and real-world application, leaving learners overwhelmed by the complexity of topics such as encryption, attack frameworks, and network security protocols. CodeForesight leverages large language models

(LLMs) and advanced prompt engineering strategies to offer real-time assistance by generating theoretical explanations and visual representations like flowcharts, data flow diagrams, and system architectures. This integration enables students to explore cybersecurity topics in a more engaging and digestible format. The platform emphasizes strengthening foundational conceptual knowledge—a critical skill set for students in cybersecurity and software development. The platform offers a webbased interface that facilitates user interaction through natural language prompts. Once a prompt is entered, the system generates a Graphviz-based diagram and a corresponding textual explanation tailored to the user's learning goal. This dualmodality design fosters improved conceptual retention and supports students in developing analytical skills essential for navigating the complexities of cybersecurity. Furthermore, its cloud-based accessibility ensures seamless use across devices without the need for local installations, thus promoting an uninterrupted, guided learning experience. By offering contextualized visual and textual content, CodeForesight enhances students' capacity for independent learning and critical thinking. It bridges the gap between traditional textbook-based education and the demands of contemporary technical problem-solving. As AI continues to influence education and cybersecurity, CodeForesight aims to empower future professionals with tools that adapt to evolving learning needs. The system architecture of CodeForesight illustrates the complete workflow from user input on the web interface to the model-generated response, as shown in the figure.

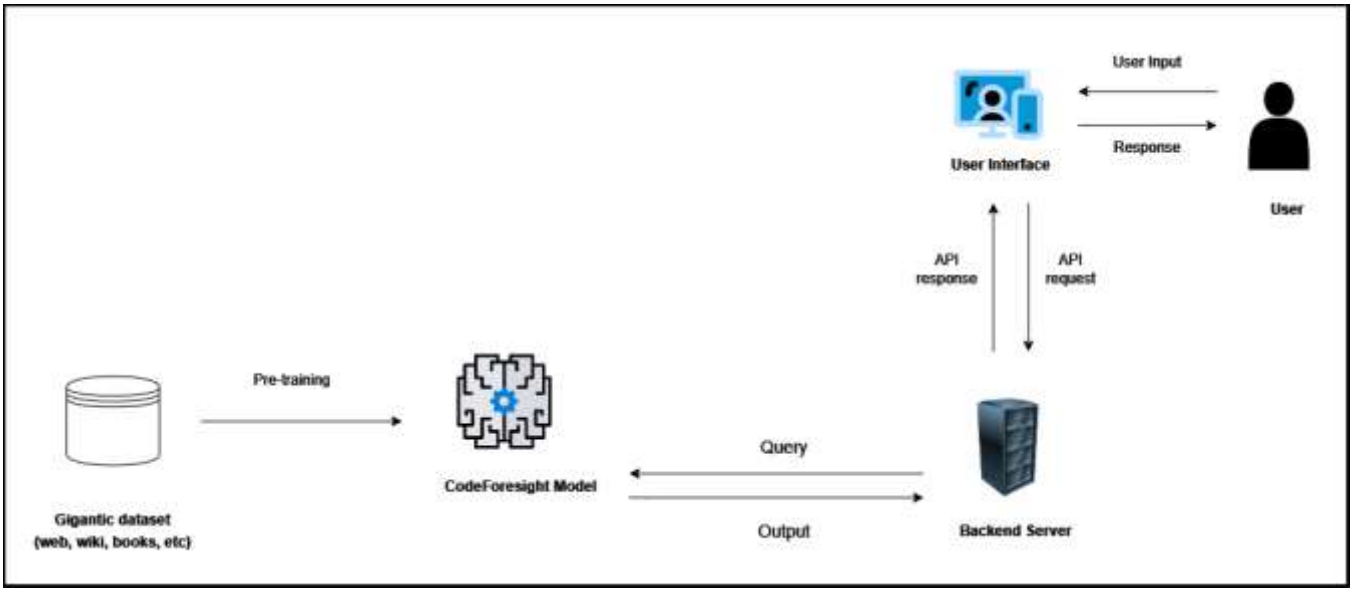

Fig. 1. CodeForesight System Architecture

2. Related Work

Recent advances in educational technology have seen large language models (LLMs) increasingly used to automate responses to student inquiries and to generate source code. Despite these successes, the majority of such tools remain focused on text-based assistance and do not adequately support visual learning—which is crucial for mastering intricate topics like cybersecurity. The LLaMA (Large Language Model Meta AI) series represents a notable open alternative to closed-source, largescale models. By striking a pragmatic balance between computational efficiency and output quality, LLaMA models are well suited for adaptation to specialized domains. In CodeForesight, we employ a fine-tuned variant of LLaMA that not only produces textual explanations but also generates diagram code in response to user prompts. To

make fine-tuning more practical on modest hardware, we integrate parameterefficient techniques such as Low-Rank Adaptation (LoRA) and quantized low-rank adaptation (QLoRA) during our training pipeline. Prompt-driven coding frameworks have illustrated how carefully designed prompts can accelerate the development of software snippets. However, these frameworks generally overlook the incorporation of visual elements and lack deep domain awareness. While diffusion-based architectures have been trialed for generating illustrative content, their potential for creating instructional diagrams tailored to pedagogy remains largely untapped. To date, few AI-driven educational platforms deliver an integrated experience combining natural language generation, structured diagram creation, and domain-specific expertise. CodeForesight fills this void by fusing NLP capabilities with automated diagram synthesis, yielding a dual-modality learning environment customized for cybersecurity education. This integrated approach empowers learners with instantaneous textual and visual guidance, fostering deeper comprehension and facilitating self-directed study.

3. Methodology

CodeForesight is designed as an intelligent, web-based educational assistant that leverages artificial intelligence to facilitate the learning of complex cybersecurity concepts. It does so by interpreting natural language prompts submitted by users and generating two synchronized outputs: a theoretical explanation and a corresponding visual representation in the form of a diagram. The system's architecture integrates user interaction, prompt processing, model inference, visual rendering, and data management into a seamless and efficient workflow. The following subsections describe the core components and implementation strategies of the CodeForesight platform.

3.1. Dataset used in the CodeForesight

The accuracy and relevance of CodeForesight's responses are underpinned by a meticulously curated dataset tailored to cybersecurity education. This dataset was specifically designed to reflect real-world cybersecurity practices and academic frameworks. Each data instance consists of three components: a natural language Prompt, the corresponding Graphviz Code, and a detailed Explanation. The dataset covers a wide spectrum of topics such as Security Operations Center (SOC) workflows, threat intelligence lifecycles, risk assessment models, Zero Trust architecture, incident response procedures, disaster recovery frameworks, and more. Every entry was manually crafted to ensure high educational value, contextual clarity, and diagrammatic accuracy. This dual-layered structure enabled the model to simultaneously learn how to explain and visualize cybersecurity concepts in a consistent and pedagogically effective manner.

Table 1. CodeForesight Dataset for Finetune

No.	Prompt	Graphviz Code	Explanation
1	Visualize the Zero Trust Security Model	digraph ZeroTrust { rankdir=TB; node [shape=box]; Identity_Verification -> Access_Policy; Access_Policy -> Resource_Protection; Resource_Protection -> Network_Segmentation;	This diagram shows Zero Trust components: Identity Verification checks all access attempts, Access Policy defines permissions, Resource Protection implements controls, Network Segmentation isolates resources, and Continuous Monitoring ensures security.

		Network_Segmentation -> Continuous_Monitoring; }	
2	Map Security Operations Center Workflow	digraph SOC { Alert_Generation -> Triage; Triage -> Investigation; Investigation -> Response; Response -> Documentation; }	Illustrates SOC workflow from alert generation through triage, investigation, incident response, to final documentation. Shows critical path for security incident handling.
3	Create Incident Response Framework	digraph IR { Preparation -> Detection; Detection -> Analysis; Analysis -> Containment; Containment -> Eradication; Eradication -> Recovery; }	Shows incident response lifecycle: Preparation establishes procedures, Detection identifies incidents, Analysis evaluates impact, Containment limits damage, Eradication removes threats, Recovery restores systems.
4	Visualize Data Loss Prevention Architecture	digraph DLP { Network -> Endpoint; Endpoint -> Storage; Storage -> Classification; Classification -> Policy; }	Demonstrates DLP components: Network monitors data in transit, Endpoint protects device data, Storage secures stored data, Classification categorizes data, Policy enforces protection rules.
5	Map Threat Intelligence Process	digraph TI { Collection -> Processing; Processing -> Analysis; Analysis -> Dissemination; Dissemination -> Action; }	Shows threat intelligence workflow: Collection gathers data, Processing normalizes information, Analysis derives insights, Dissemination shares intelligence, Action implements defenses.

3.2. Model Initialization

For the core architecture of CodeForesight, we selected Meta's LLaMA 3 (Large Language Model Meta AI) as the foundational pre-trained model because of its cutting-edge capabilities in natural language understanding and generation. LLaMA 3 introduces notable enhancements over earlier versions, including improved scalability, deeper contextual awareness, and more efficient token utilization—features essential for developing an intelligent AI assistant tailored to cybersecurity education. The decision to use LLaMA 3 was motivated by its openly accessible weights, solid transformer-based design, and strong generalization across a wide range of tasks. It supports processing of multilingual inputs, extended context windows, and detailed reasoning—capabilities that align perfectly with CodeForesight's requirements to comprehend technical prompts, generate code examples, deliver theoretical explanations, and provide visual elements like diagrams or system architecture flows. Furthermore, LLaMA 3 integrates well with popular opensource ecosystems such as Hugging Face's Transformers and PyTorch frameworks, allowing smooth incorporation into our development workflow. Its flexible and modular architecture enables domain-specific customization for cybersecurity, where accurate terminology, structured output, and precise response generation are critical. By grounding our project in LLaMA 3, we established a robust base model that supports efficient fine-tuning and maintains high-quality outputs across specialized and complex tasks.

3.3. Fine-Tuning Setup and Process

The LLaMA 3 model underwent fine-tuning to tailor its performance specifically for the objectives of CodeForesight—namely, producing precise theoretical content alongside Graphviz-based diagram code

for cybersecurity topics. This adaptation aimed to empower the model to interpret domainfocused prompts and generate educational materials comprising both explanatory text and code that creates relevant visualizations, thus enhancing learners' conceptual understanding.

Training Environment Setup:

To handle the demanding compute requirements of training a large language model, we utilized a high-performance computing setup featuring NVIDIA A100 GPUs and multi-core CPUs. The training environment was constructed using Python, leveraging established libraries such as PyTorch and Hugging Face's Transformers, which facilitated tokenizer integration, model configuration, training orchestration, and evaluation. For consistency, scalability, and ease of deployment, the environment was containerized using Docker and hosted on GPU-enabled cloud platforms. This infrastructure ensured optimal utilization of resources and enabled flexible experimentation with different training parameters and dataset iterations.

Fine-Tuning Strategy:

The fine-tuning methodology employed a supervised learning paradigm, training LLaMA 3 on a carefully compiled dataset consisting of user prompts paired with corresponding cybersecurity theoretical explanations and Graphviz code snippets. These code snippets were designed to generate instructional diagrams such as flowcharts, attack trees, protocol diagrams, and network models relevant to cybersecurity concepts queried by users. This deliberate and controlled fine-tuning process enabled the model to specialize in producing both accurate theoretical content and executable diagram code, effectively transforming LLaMA 3 into a dependable educational assistant within the CodeForesight platform.

3.4. Evaluation and Validation

To ensure the quality and reliability of the fine-tuned LLaMA 3 model, a multi-metric evaluation approach was employed. This process assessed the model's ability to accurately interpret cybersecurity-related prompts and generate both meaningful textual explanations and corresponding Graphviz code for educational diagrams. The evaluation phase focused on three main dimensions: visual relevance, semantic fidelity, and educational value.

- **Average CLIP Score**: Visual Relevance to Prompt: The CLIP (Contrastive Language–Image Pretraining) model was used to compute the Average CLIP Score, which measured the alignment between the cybersecurity-related prompts and the images generated from the model's Graphviz output. This metric was crucial in evaluating how well the visual representations reflected the intended cybersecurity concept described in the user's prompt. A high CLIP score indicated that the generated diagram was contextually relevant and accurately depicted structures such as firewalls, attack vectors, network architectures, or encryption flows, thereby confirming the visual clarity and appropriateness of the response.
- **Average SBERTScore**: Semantic Closeness of Textual Explanation: To evaluate the textual portion of the output, we used SBERTScore, which leverages Sentence-BERT to assess the semantic similarity between the generated explanation and the original input prompt. This metric was particularly useful for measuring how effectively the model understood and articulated complex cybersecurity topics. Unlike traditional surface-level metrics such as BLEU or ROUGE, SBERTScore captures deeper sentence-level contextual and semantic alignment. This makes it especially suitable for domain-specific tasks like cybersecurity, where conceptual accuracy and the correct use of technical vocabulary are crucial. A higher SBERTScore indicated that the generated content was semantically faithful to the prompt, thus maintaining conceptual integrity.

- **Average Human Score**: Educational Usefulness: To assess the real-world impact of the system, particularly its effectiveness in educational settings, we conducted a manual evaluation by cybersecurity instructors and final-year engineering students. They rated each output based on three criteria: accuracy of explanation, diagram clarity, and overall usefulness for learning. This qualitative metric was aggregated into an Average Human Score, which offered insights into the model's ability to generate content that is not only technically correct but also pedagogically valuable. This human-centered evaluation helped validate the system's role as an AI-powered learning assistant.

3.5. Web-Based Deployment

To ensure accessibility and usability of the CodeForesight system, a complete web-based deployment architecture was implemented. The system is divided into three main components: the frontend, backend, and database, all integrated via secure RESTful APIs.

Frontend:

Built using Flutter, the frontend provides a user-friendly interface for submitting prompts and viewing AIgenerated explanations and diagrams.

Backend:

Implemented in Python, the backend follows a structured three-phase processing pipeline:

- Prompt Classification: Incoming prompts are categorized into types such as flowchart generation, dataflow diagram, system architecture explanation, or concept visualization.
- Theoretical Explanation Generation: A domain-specific textual explanation is generated to describe the cybersecurity concept in detail.
- Diagram Generation: Corresponding Graphviz Dot code is generated, rendered into an image, and encoded into Base64 format for seamless delivery to the frontend.

Database (MongoDB):

Four interconnected collections manage application data:

- Users: Stores user information including hashed credentials and account status.
- Prompts: Logs user-submitted queries with metadata such as category and timestamp.
- Responses: Contains AI-generated explanations and model metadata.
- Images: Stores Dot code and Base64-encoded diagrams for frontend display.

API Integration: RESTful APIs, developed using Flask or FastAPI, enable secure and efficient communication between frontend and backend. Endpoints support user authentication, prompt submission, response retrieval, and profile access. JWT-based token authentication and middleware ensure secure and scalable operations. This modular and efficient deployment design enables realtime interaction, secure data handling, and effective delivery of AI-driven educational content in the field of cybersecurity

4. Results

To begin using the CodeForesight AI Assistant, users must first create an account on the mobile or web-based platform. The registration process includes entering basic credentials such as name, email, password, gender, and mobile number. Upon submission, the user details are stored securely in the backend database. This step ensures identity verification and personalization of user data within the application. After successfully registering, users can log in using their registered credentials. The login system validates the input email or username and password against the stored values in the database. If the credentials are correct and the user is active, access is granted to the dashboard. If any mismatch occurs or the user is deactivated, appropriate alerts are shown to the user, preventing unauthorized access.

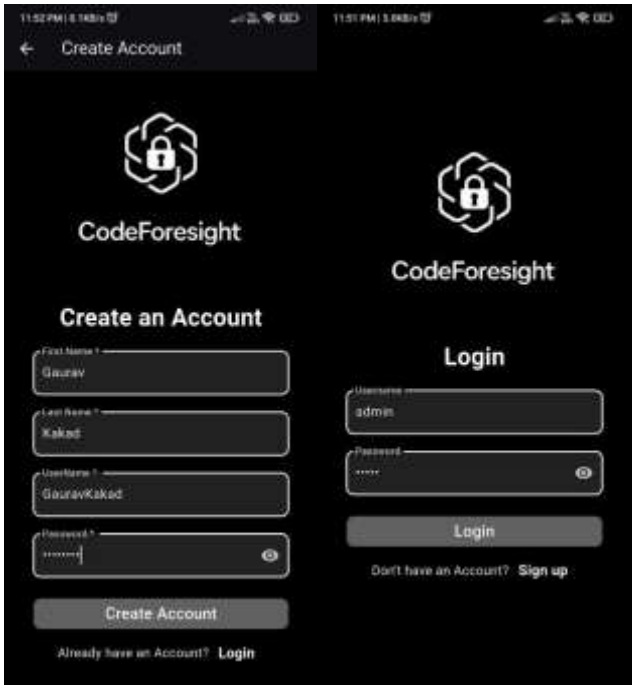

Fig 2. CodeForesight Application Sign – up and Login Page

Once logged in, the user is directed to the dashboard interface. Here, the user can view their past interactions through chat history. A prompt input box is provided where users can type their queries or coding-related prompts. The clean and responsive layout of the dashboard supports real-time communication with the CodeForesight assistant. The user enters a specific query or task into the prompt box. These prompts usually relate to cybersecurity topics and may include questions like: “Visualize the Zero Trust Security Model”}. This allows learners or developers to gain structured visualizations or code explanations related to cybersecurity architectures and principles.

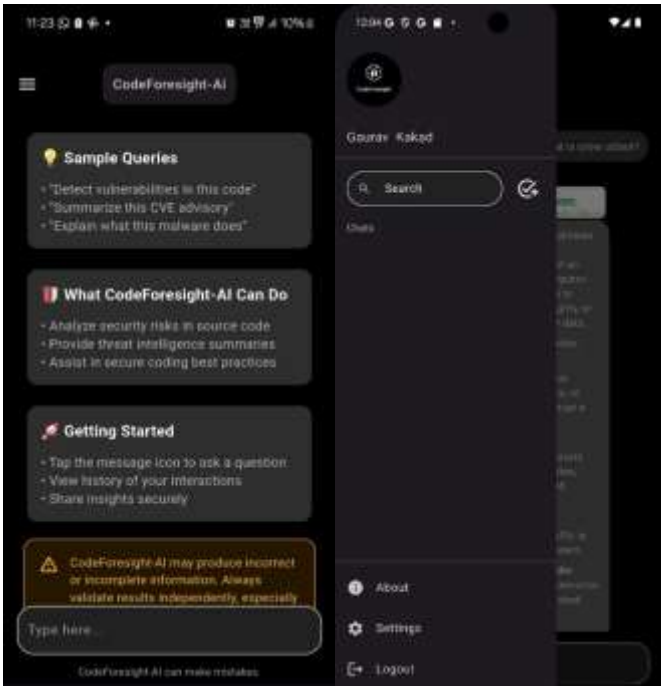

Fig 3. CodeForesight Application User Dashboard

After the user submits the prompt, the request is processed through the backend, the AI assistant interprets the request contextually, queries the knowledge base, and generates a detailed response which is then returned to the frontend and displayed in a chat-style format. This process occurs seamlessly, to ensure a smooth user experience.

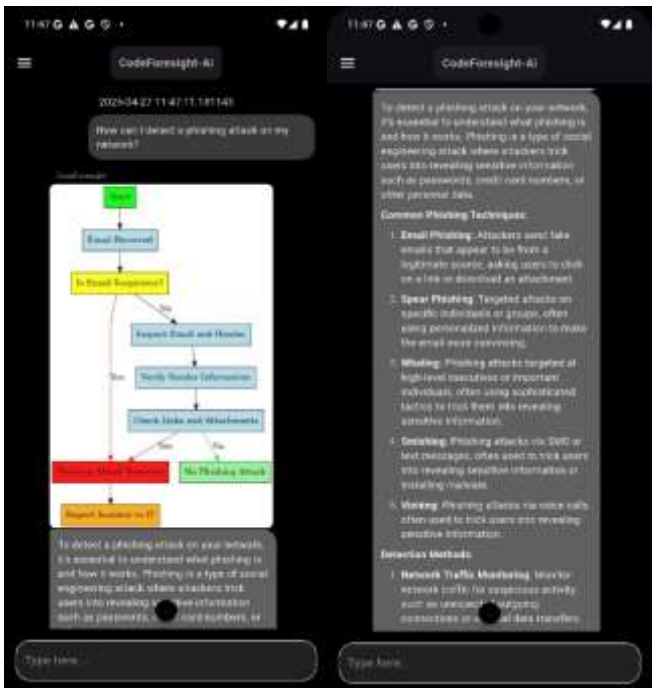

Fig 4. CodeForesight Application Prompt Processing and AI Response

5. Performance Evaluation

The effectiveness of the CodeForesight: AI-Powered Learning and Coding Assistant was evaluated using a combination of quantitative and qualitative metrics. The focus was to determine how well the AI-generated content—both visual and textual—aligns with user prompts, and whether it supports meaningful learning in the domain of cybersecurity. The following evaluation metrics were used:

- **CLIP Score**
- **SBERTScore**
- **Human Evaluation**

Model	CLIP Score	SBERTScore	Human Score	Notes
CodeForesight	0.305	0.673	0.769	Image and Text generated
ChatGPT (DALL.E)	0.322	0.641	0.431	Image and Text generated
Stable Diffusion	0.278	-	0.22	Only Image generated

Table 2. Comparison of Model Outputs on cybersecurity prompts using CLIP, SBERT and Human evaluation

These offer a well-rounded assessment of the assistant's ability to generate relevant, meaningful, and educationally valuable content. This multidimensional evaluation ensures that the system is not only technically accurate but also pedagogically effective in the domain of cybersecurity.

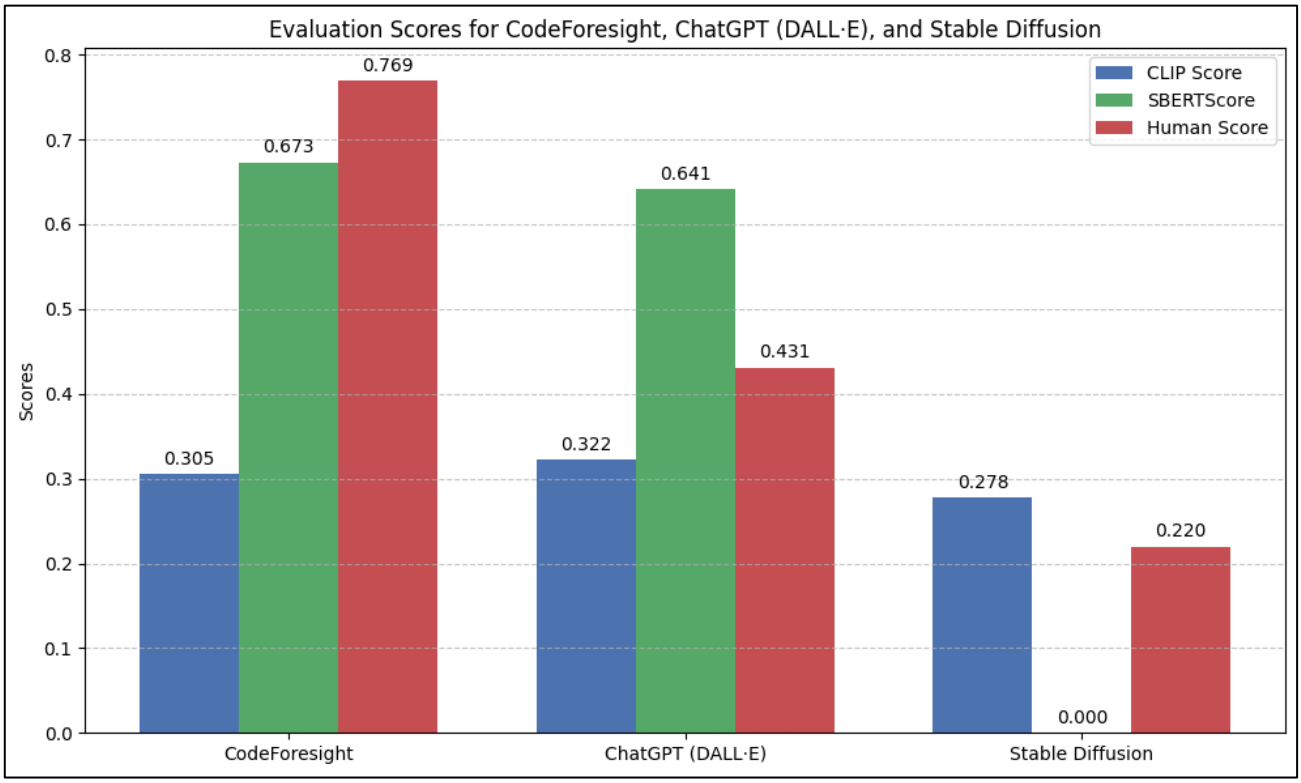

Fig 5. Model Evaluation based on CLIP, SBERT and Human evaluation scores

From the table and visualization:

- **CLIP Score: ChatGPT (DALL·E)** is slightly ahead.
- **SBERTScore: CodeForesight** has the highest.
- **Human Score: CodeForesight** performs best, significantly better than others.

Based on a holistic comparison of CLIP Score (visual-text alignment), SBERTScore (semantic explanation quality), and Human Evaluation (perceived performance), **CodeForesight** emerges as the best-performing model overall for generating both image and textual explanations on cybersecurity prompts.

6. Conclusion

In this research, we presented CodeForesight, an AI-powered educational assistant designed to assist users in understanding complex cybersecurity concepts through both textual explanations and visual diagrams. By fine-tuning a pre-trained LLaMA model and integrating it with a robust web-based system, CodeForesight generates accurate, domain-specific content, ensuring that users receive both theoretical insights and visual representations. The implementation of a secure, scalable API and a well-structured backend ensures seamless interaction between the frontend and model, enhancing the overall user experience. Evaluation metrics, including CLIP scores, SBERTScore, and human assessments, demonstrate the system's effectiveness in generating educationally valuable content. This work paves the

way for future advancements in AI-driven educational tools for cybersecurity, with potential applications in automated learning platforms and online training resources.

7. Acknowledgements

We would like to acknowledge all the teachers and friends who helped and assisted us throughout our project work. First of all, we would like to express our sincere gratitude to our respected guide Mrs. P. R. Patil for her continuous guidance, support, and encouragement. This project would not have been possible without her insightful suggestions and valuable ideas. Furthermore, we are deeply thankful to Dr. R. J. Patil, Principal of Trinity Academy of Engineering, for the continuous support during our project work. We also extend our heartfelt thanks to all the faculty members of the Information Technology Department at Trinity Academy of Engineering, Pune, for their cooperation and support. We acknowledge the use of the LLaMA model by Meta AI for generating code snippets and visual representations to support the development of the CodeForesight platform, with all responsibility for the final content resting with the authors. We would like to thank our parents for their unwavering support and inspiration throughout this journey, as well as our friends for their suggestions, help, and constant encouragement. Finally, we would like to acknowledge the blessings of the Almighty, whose grace kept our morale high during challenging times.

References

1. Youjia L., Jianjun S., Zheng Z., “An Approach for Rapid Source Code Development Based on ChatGPT and Prompt Engineering”, IEEE Access, 2024, 12, 53074–53087. https://doi.org/10.1109/access.2024.3385682
2. Sunny A., “A Review on Various Methods of Cryptography for Cyber Security”, Journal of Algebraic Statistics, May 2022. https://publishoa.com/index.php/journal/article/view/1353
3. Shuaib A.W., Areej F.M., Aun Y., Ramesh K., Farhan B.S., “Encryption Techniques and Algorithms to Combat Cybersecurity Attacks: A Review”, VAWKUM Transactions on Computer Sciences, Jun. 2023, 11 (1), 295–305. https://doi.org/10.21015/vtcs.v11i1.1521
4. Vijayaraghavan M., Chandra M., Imad A., et al., “AI-Assisted Code Authoring at Scale: Fine-Tuning, Deploying, and Mixed Methods Evaluation”, Proceedings of the ACM on Software Engineering, Jul. 2024, 1 (FSE), 1066–1085. https://doi.org/10.1145/3643774
5. Y. Wang, H. Le, A.D. Gotmare, N. Bui, J. Li, S. Hoi, “CodeT5+: Open Code Large Language Models for Code Understanding and Generation”, GitHub, May 2023. https://github.com/salesforce/CodeT5-plus
6. Mohammed L.S., Shafayat H.M., Maisha R.M., Sourov J., Joanna, “An Empirical Study of Code Smells in Transformer-based Code Generation Techniques”, 2022 IEEE 22nd International Working Conference on Source Code Analysis and Manipulation (SCAM), Oct. 2022. https://doi.org/10.1109/scam55253.2022.00014
7. Chanda H., Enda F., Paul C., Kieran F., Deepak Y., “A Comparative Study of Intent Classification Performance in Truncated Consumer Communication using GPT-Neo and GPT-2”, 2023 International Conference on Emerging Techniques in Computational Intelligence (ICETCI), Sep. 2023. https://doi.org/10.1109/icetci58599.2023.10331337

8. Toka A.M., Mohamed H.K., Ahmed B.E., Ahmed S.I., "A Proposed Model for Distinguishing Between Human-Based and ChatGPT Content in Scientific Articles", IEEE, Aug. 22, 2024. https://ieeexplore.ieee.org/document/10564718
9. Tingshuai C., Ye Y., Bingyang Y., "Application of Prompt Engineering in AIGC — Taking Stable Diffusion as an Example", 2024 4th International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Jun. 2024, 465–469. https://doi.org/10.1109/mlise62164.2024.10674412
10. Michael C., Muhammad R., William S., H. Lucky, J.V. Moniaga, "Accuracy and Fidelity Comparison of Luna and DALL-E 2 Diffusion-Based Image Generation Systems", IEEE Xplore, 2024. https://ieeexplore.ieee.org/document/10552890
11. Donghao H., Zhenda H., Zhaoxia W., "Performance Analysis of Llama 2 Among Other LLMs", 2024 IEEE Conference on Artificial Intelligence (CAI). https://doi.org/10.1109/CAI59869.2024.00108
12. Xiao-Yang L., Jie Z., Guoxuan W., Wei T., Ahmed W., "Efficient Pretraining and Finetuning of Quantized LLMs with Low-Rank Structure", 2024 IEEE Conference on Computational Intelligence. https://doi.org/10.1109/CIC58760.2024.00098
13. Avik P., Om S., Mallika A., Shek D.S., Anupam T., "Performance Analysis of LoRA Finetuning Llama-2", 2023 7th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech), Dec. 2023. https://doi.org/10.1109/iementech60402.2023.10423400
14. Junyi L., Lei Y., Xiaojia L., Li Y., Chun Z., "LLaMA-Reviewer: Advancing Code Review Automation with Large Language Models through Parameter-Efficient Fine-Tuning", 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), Oct. 2023. https://doi.org/10.1109/issre59848.2023.00026
15. Harshil T., A. Manimaran, "Comprehensive Examination of Instruction-Based Language Models: A Comparative Analysis of Mistral-7B and Llama-2-7B", IEEE International Conference on Emerging Research in Computational Science – ICERCS'23, Dec. 2023. https://doi.org/10.1109/icercs57948.2023.10434081
16. Hugo T., Thibaut L., Gautier I., et al., "LLaMA: Open and Efficient Foundation Language Models", arXiv preprint, Feb. 2023. https://arxiv.org/abs/2302.13971
17. Ashish V., Noam S., Niki P., et al., "Attention Is All You Need", arXiv preprint, Jun. 2017. https://arxiv.org/abs/1706.03762
18. Edward H., Yelong S., Phillip W., et al., "LoRA: Low-Rank Adaptation of Large Language Models", arXiv preprint, Oct. 2021. https://arxiv.org/abs/2106.09685
19. Tim D., Artidoro P., Luke Z., "QLoRA: Efficient Finetuning of Quantized LLMs", arXiv preprint, May 2023. https://arxiv.org/abs/2305.14314