Building Code Explainer Using LLM

Dipali Kalamdhad[1], Harsha Bambal[2], Shital Bansod[3], Neha Khawse[4], Gauri Khode[5], Prof. Kanchan Raipure[6]

Computer Science and Engineering, SBITM, Betul (M.P).

Abstract

The Code Explainer project seeks to develop an AI system for translating complex code snippets into human-readable explanations. This project addresses the demand for simplified code comprehension tools, targeting developers, students, and non-technical stakeholders. The Code Explainer platform utilizes Natural Language Processing (NLP) and Language Model (LLM) to enhance explanation quality and deliver a seamless user experience through a user-friendly interface. The Code Explainer project's emphasis on functionality and user-centric design concepts is essential to its success. The UI of the platform is designed to be both simple and intuitive, making it easy for users to submit code samples and obtain detailed explanations with no effort. In addition to its core functionality, the Code Explainer platform incorporates advanced features such as the Save Option. This feature empowers users to save the generated explanations for future reference, allowing for convenient review or sharing. Furthermore, the Saved Explanations Page Module provides users with a dedicated space to manage their saved explanations, enabling them to view, edit, or delete entries as needed. The platform also boasts a powerful Search Functionality, enriching the user experience by providing a robust tool to discover saved explanations. Users can enter search queries based on specific keywords or code snippets into the search bar, and the system intelligently analyzes the input to retrieve relevant explanations from the platform's database.

Keywords: Code Explainer, Code Explanation Generator, LLM, Html, CSS, ReactJs, NodeJs, Firebase.

I. INTRODUCTION

A Code Explainer is a AI-based system designed to analyze and explain code snippets, scripts, or complete programs in a human-readable and understandable manner. It serves as a valuable tool for developers, students, and non-technical stakeholders who may not be proficient in coding but need to understand the functionality and logic of a piece of code. The Code Explainer platform is a tool enabling users to receive clear and concise explanations for code segments. This user-friendly tool interprets various programming languages and provides detailed insights into the functionality of the code. Through a fusion of cutting-edge technologies and intuitive design principles, this project endeavors to revolutionize the way we perceive and explain complex code snippets. The primary objective is clear: to develop an AI system that not only interprets code but also generates easily understandable and contextually relevant explanations tailored to the diverse needs of its users. Central to the success of the Code Explainer project is its emphasis on user-centric design principles and functionality. The platform boasts an intuitive and user-friendly interface, ensuring that users can effortlessly input their code snippets and retrieve comprehensive explanations with minimal friction. In addition to its core functionality, the Code Explainer platform incorporates advanced features such as the Save Option. The Save Option allows users to save the generated explanations for future reference. After generating an explanation, users can choose to save it for later review. Furthermore, the Saved Explanations Page Module provides users with a dedicated space to manage their saved explanations, enabling them to view explanation as needed. The platform also boasts a powerful Search Functionality, enriching the user experience by providing a robust tool to explore and discover saved explanations. Users can enter search queries based on specific keywords or code snippets into the search bar, and the system intelligently analyzes the input to retrieve relevant explanations from the platform's database. The Code Explainer project aims to revolutionize the way we comprehend and explain complex code snippets. The primary goal is to develop an Artificial

Intelligence (AI) system that not only interprets code but generates easily understandable and contextually relevant explanations for diverse audiences.

Objective :-
The aim is to develop a Code Explainer with improved capabilities. The system will incorporate advanced techniques and technologies to enhance the performance and user experience. Here are some key objectives:
1. Enhanced Contextual Understanding: The proposed system will utilize natural language understanding to achieve better contextual understanding and provide more accurate and relevant responses.
2. Enhanced User Experience: Users will have a more customized and interactive experience with the Code Explainer. Users will be able to communicate with the system without difficulty.
3. User-Friendly Interface: Provide an intuitive and easy-to-use interface that enables users to effortlessly generate explanation of the code.
4. Text Input Flexibility: The platform will offer flexibility in text input, supporting various forms of code representation including plain text, code snippets. The system will accommodate their input preferences seamlessly, ensuring a smooth and intuitive user experience.
5. Save Option: The Save Option allows users to save the generated explanations for future reference. After generating an explanation, users can choose to save it for later review.
6. Saved Explanations Page: The Saved Explanations Page Module displays the list of saved explanations along with the corresponding code snippets. Users can view the saved explanations as needed.
7. Search Functionality: The Search Functionality enriches the user experience by providing a powerful tool to search saved explanations. This enables users to search for explanations based on specific keywords or code snippets. Users can enter search queries into the search bar, and the system intelligently analyses the input to retrieve relevant explanations from the platform's database.

II. PROBLEM DEFINITION

Understanding complex code snippets poses a significant challenge across diverse user backgrounds, from developers to non-technical stakeholders. The Code Explainer project aims to tackle this challenge by developing an Artificial Intelligence (AI) system capable of interpreting code inputs and generating easily understandable and contextually relevant explanations. The project seeks to bridge these gaps and enhance code comprehension for effective collaboration, learning, and communication in the programming domain. The Code Explainer project aims to tackle the challenge of transforming complex code snippets into easily understandable and contextually relevant explanations. The primary goal is to develop an AI system that comprehends code inputs and generates human-readable explanations.

III. LITERATURE REVIEW

“Exploring Large Language Models for Code Explanation” (Bhattacharya et al., 2023) stands as a foundational work in the field. This research investigates the effectiveness of LLMs in generating natural language explanations for code snippets. The study demonstrates that LLMs specifically trained on code-related data outperform generic LLMs in this task. This finding highlights the importance of domain-specific training for achieving superior code explanation capabilities. [1]

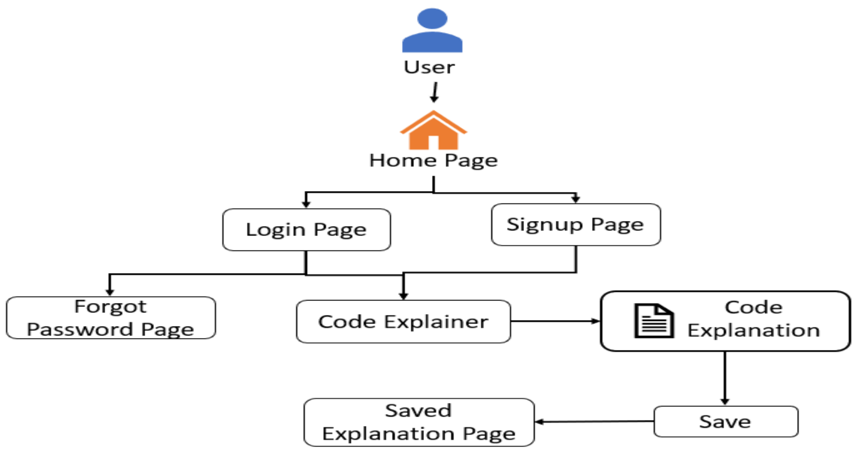
"GPTutor: A ChatGPT-powered Programming Tool for Code Explanation," Chen et al. (2023) introduce a novel system that utilizes the capabilities of the GPT-3 language model to generate natural language explanations for code snippets. The authors propose an approach that combines code parsing, code embedding, and language model querying to produce human-understandable descriptions of the functionalities and logic behind the provided code. Their work demonstrates the effectiveness of leveraging large language models in building practical educational tools for programming, facilitating the learning process for students and novice developers. [2]

"Llama 2: Open Foundation Models" by Touvron et al. (2023) introduces Llama 2, a new generation of open-source large language models developed by Anthropic. Llama 2 builds upon the original Llama model and incorporates several advancements, such as improved performance on a wide range of tasks, enhanced safety and robustness, and the ability to be fine-tuned for specific applications. The authors demonstrate the potential of Llama 2 to serve as a powerful foundation for various natural language processing tasks, including code understanding and explanation.[3]

"A Survey of Large Language Models" by Zhao et al. (2022) presents a comprehensive survey of large language models, covering their history, architecture, training techniques, and applications. The authors explore the evolution of these models, from the early successes of GPT-3 to the more recent advancements in areas like multilingual and multimodal language understanding. This work provides a solid foundation for understanding the capabilities and limitations of large language models, which is crucial for designing effective code explainer systems.[4]

"A Comprehensive Overview of Large Language Models" by Naveed et al. (2022) provides a thorough review of the state-of-the-art in large language models. The authors discuss the architectural advancements, training techniques, and applications of these models, highlighting their remarkable achievements in tasks like language generation, question answering, and code generation. This work underscores the versatility and potential of large language models, which can be leveraged for developing advanced code explainer systems.[5]

IV. SYSTEM ARCHITECTURE AND METHODOLOGY



1. Home Page: Our web platform's primary entrance point is the Home Page Module. It features an inviting interface that offers users a glimpse into the myriad features and functionalities available on our platform. Users can explore the features of our code explanation tool by starting from this page.

2. Login Page: The Login Page Module verifies the identity of visitors to the website using their login credentials, which include an email address and password. Users are able to access different parts of the website after providing legitimate login credentials. The "Forgot Password" button allows users to access the Forgot Password Page and begin the password reset procedure in the event that they forget their password. This module ensures secure access to the platform while providing a convenient solution for password recovery if needed.

3. Signup Page: The Signup Page Module enables new users to register for an account. User data, including login credentials, are securely stored in our database with passwords hashed for added security. Each user is assigned a unique identifier upon registration, ensuring system integrity and preventing unauthorized access.

4. Forgot Password Page: The Forgot Password Page Module assists users in recovering their account credentials in case they forget them. By providing necessary verification steps, users can reset their passwords securely, maintaining access to their accounts.

5. Code Explainer Page: The Code Explainer Page Module is the core feature of our platform. Here, users input their code snippet to generate an explanation. The page offers a user-friendly interface for seamless code understanding.

6. Save Option: The Save Option allows users to save the generated explanations for future reference. After generating an explanation, users can choose to save it for later review.

7. Saved Explanations Page: The Saved Explanations Page Module displays the list of saved explanations along with the corresponding code snippets. Users can view the saved explanations as needed.

8. Search Functionality: The Search Functionality enriches the user experience by providing a powerful tool to search saved explanations. This enables users to search for explanations based on specific keywords or code snippets. Users can enter search queries into the search bar, and the system intelligently analyses the input to retrieve relevant explanations from the platform's database.

V. SYSTEM ANALYSIS AND DESIGN

DFD Diagrams –

1. Zero Level DFD :-

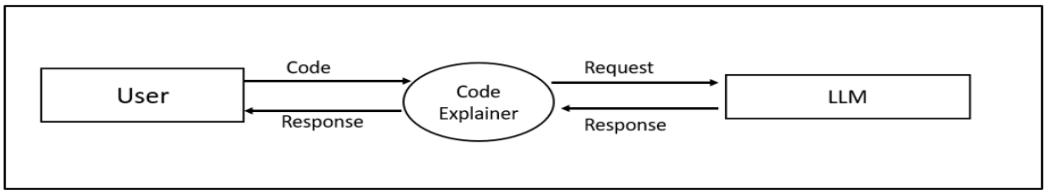


Fig: Zero Level DFD

2. First Level DFD :–

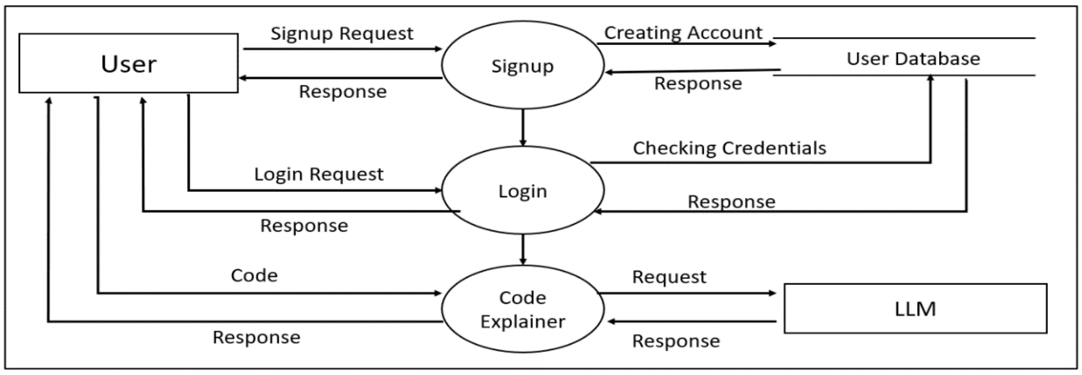


Fig: First Level DFD

3. Second Level DFD:-

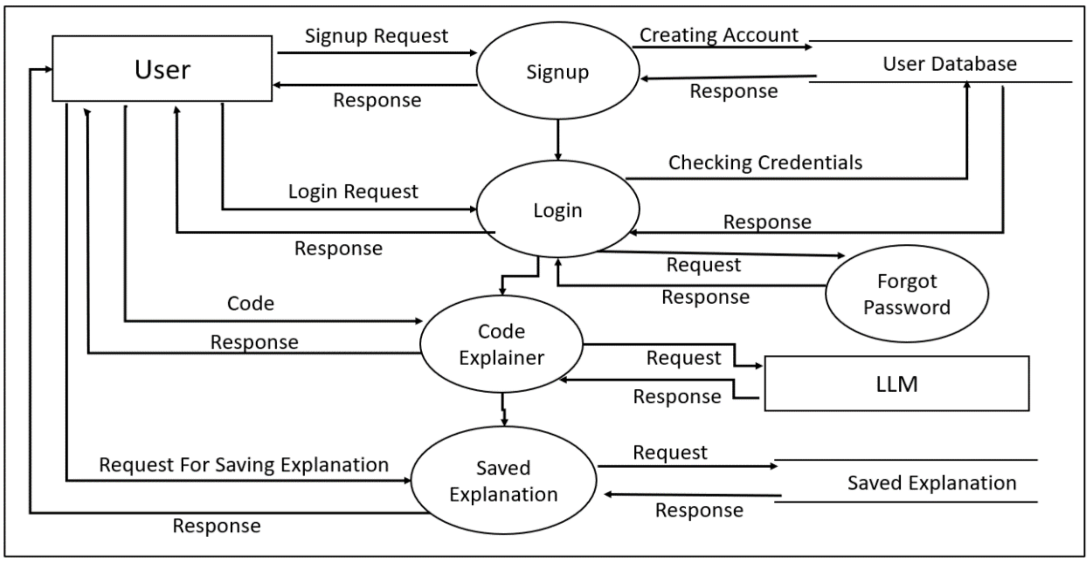


Fig: Second Level DFD

VI. TECHNOLOGIES

1. HTML:– HTML, short for Hyper Text Markup Language, serves as the fundamental markup language for crafting documents meant to be showcased in a web browser. It seamlessly collaborates with supplementary technologies like Cascading Style Sheets (CSS) and dynamic scripting languages such as JavaScript.

2. CSS :– CSS, short for Cascading Style Sheets, is a language for styling web documents, like HTML. It excels in separating presentation—layout, colours, and fonts—from content, enhancing accessibility and flexibility. By specifying CSS in a separate file, it enables consistent formatting across multiple pages, reducing complexity and optimizing page load speed through caching.

3. JavaScript :– JavaScript is a dynamic computer programming language. It is lightweight an most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming languages with object-oriented capabilities.

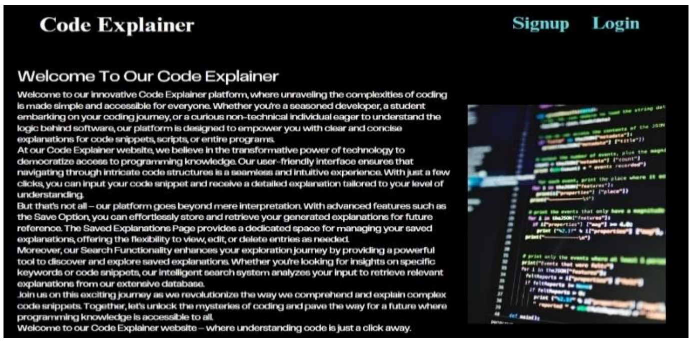
4. ReactJS :- React is a JavaScript library designed for building dynamic and interactive user interfaces (UIs) in web applications. Developed by Facebook, it follows a component-based architecture. With React, developers can create reusable UI components that manage their own states, and the library efficiently updates the UI when data changes. This "virtual DOM" approach optimizes performance, making react a popular choice for crafting modern, responsive, and efficient front-end experiences.

5. Node.js :- Node.js is a cross-platform JavaScript runtime environment that executes JavaScript code on the server side, allowing developers to build scalable and high-performance server-side applications using the same language as the front end.Node.js provides a non-blocking, event-driven architecture, which makes it highly scalable and efficient for handling concurrent requests.

6. Firebase :- Firebase is a comprehensive platform for building web and mobile applications. Firebase provides various services such as database, hosting, authentication, and cloud functions. Firebase's database allows for real-time data synchronization across clients. Firebase's authentication service provides secure user authentication and authorization features.

7. LLM :- A large language model (LLM) is a type of artificial intelligence (AI) model that is able to understand and generate text. These models are designed to process and generate text, making them useful for a wide range of natural language processing (NLP) tasks such as text generation. Leveraging advanced machine learning techniques, these models have the remarkable capability to not only grasp the intricacies of language but also seamlessly produce coherent and contextually relevant text across diverse domains. By harnessing the power of neural networks and deep learning algorithms, LLMs can navigate complex linguistic structures, enabling them to excel in various natural language processing (NLP) tasks, with a primary emphasis on text generation. This versatility makes them invaluable tools for applications ranging from chatbots and language translation to content summarization and sentiment analysis, revolutionizing the way we interact with and derive insights from textual data in the realm of artificial intelligence.

VII. RESULT/OUTPUT



Welcome To Our Code Explainer

Welcome to our innovative Code Explainer platform, where unraveling the complexities of coding is made simple and accessible for everyone. Whether you're a seasoned developer, a student embarking on your coding journey, or a curious non-technical individual eager to understand the logic behind software, our platform is designed to empower you with clear and concise explanations for code snippets, scripts, or entire programs.
At our Code Explainer website, we believe in the transformative power of technology to democratize access to programming knowledge. Our user-friendly interface ensures that navigating through intricate code structures is a seamless and intuitive experience. With just a few clicks, you can input your code snippet and receive a detailed explanation tailored to your level of understanding.
But that's not all – our platform goes beyond mere interpretation. With advanced features such as the Save Option, you can effortlessly store and retrieve your generated explanations for future reference. The Saved Explanations Page provides a dedicated space for managing your saved explanations, offering the flexibility to view, edit, or delete entries as needed.
Moreover, our Search Functionality enhances your exploration journey by providing a powerful tool to discover and explore saved explanations. Whether you're looking for insights on specific keywords or code snippets, our intelligent search system analyzes your input to retrieve relevant explanations from our extensive database.
Join us on this exciting journey as we revolutionize the way we comprehend and explain complex code snippets. Together, let's unlock the mysteries of coding and pave the way for a future where programming knowledge is accessible to all.
Welcome to our Code Explainer website – where understanding code is just a click away.

Fig: Home Page

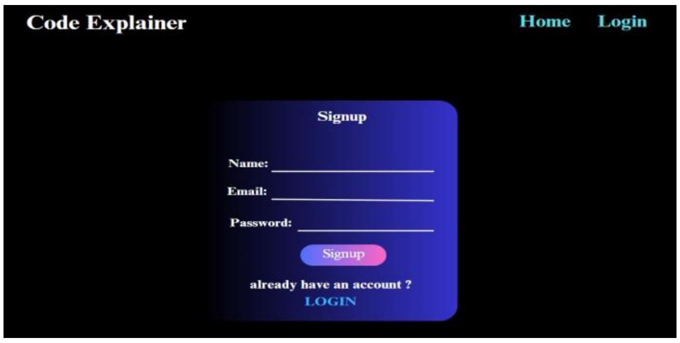


Fig: Signup Page

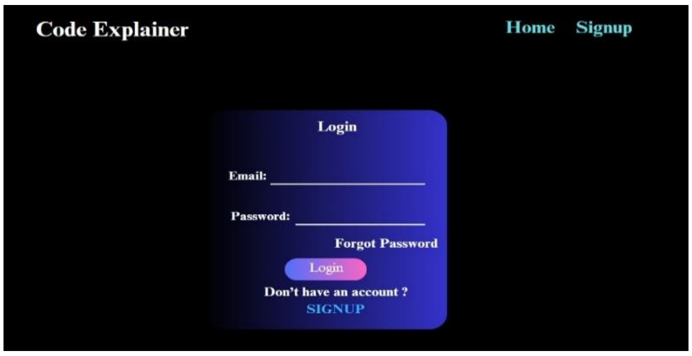


Fig: Login Page

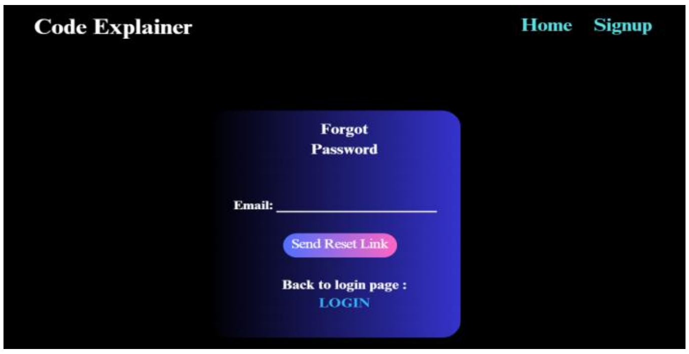
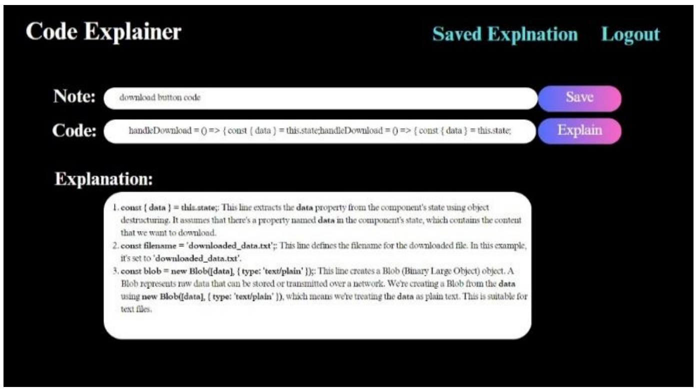


Fig: Forgot Password Page



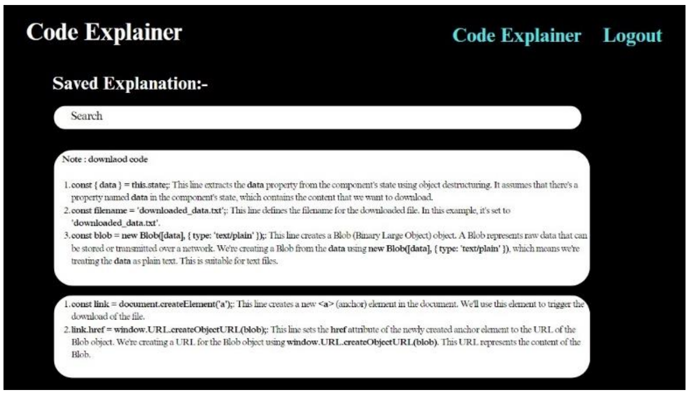
Note: download button code

Code: handleDownload = () => { const { data } = this.state;handleDownload = () => { const { data } = this.state;

Explanation:

1. **const { data } = this.state;**: This line extracts the **data** property from the component's state using object destructuring. It assumes that there's a property named **data** in the component's state, which contains the content that we want to download.
2. **const filename = 'downloaded_data.txt';**: This line defines the filename for the downloaded file. In this example, it's set to **'downloaded_data.txt'**.
3. **const blob = new Blob([data], { type: 'text/plain' });**: This line creates a Blob (Binary Large Object) object. A Blob represents raw data that can be stored or transmitted over a network. We're creating a Blob from the **data** using **new Blob([data], { type: 'text/plain' })**, which means we're treating the **data** as plain text. This is suitable for text files.

Fig: Code Explainer



Saved Explanation:-

Note : downlaod code

1. **const { data } = this.state;**: This line extracts the **data** property from the component's state using object destructuring. It assumes that there's a property named **data** in the component's state, which contains the content that we want to download.
2. **const filename = 'downloaded_data.txt';**: This line defines the filename for the downloaded file. In this example, it's set to **'downloaded_data.txt'**.
3. **const blob = new Blob([data], { type: 'text/plain' });**: This line creates a Blob (Binary Large Object) object. A Blob represents raw data that can be stored or transmitted over a network. We're creating a Blob from the **data** using **new Blob([data], { type: 'text/plain' })**, which means we're treating the **data** as plain text. This is suitable for text files.

1. **const link = document.createElement('a');**: This line creates a new <a> (anchor) element in the document. We'll use this element to trigger the download of the file.
2. **link.href = window.URL.createObjectURL(blob);**: This line sets the **href** attribute of the newly created anchor element to the URL of the Blob object. We're creating a URL for the Blob object using **window.URL.createObjectURL(blob)**. This URL represents the content of the Blob.

Fig: Saved Explanation Page

VIII. CONCLUSION

The code explainer project leverages a powerful technology stack, including HTML, CSS, JavaScript, ReactJS, Node.js, Firebase, and Large Language Model (LLM). By integrating these technologies, the project aims to create an advanced system capable of translating code into contextually relevant and human-like explanations. The integration of front-end and back-end components ensures a seamless user experience, while the utilization of LLM enhances the output response. The front-end development employs HTML, CSS, and ReactJS to create a visually appealing and user-friendly interface. HTML provides the structure of the web pages, while CSS ensures consistent styling and layout across different devices and media. ReactJS facilitates the creation of reusable UI components, enhancing code reusability and maintainability. On the back end, Node.js acts as the server-side runtime environment, enabling server-side JavaScript execution and facilitating concurrent request handling. Firebase, a comprehensive platform, offers essential services such as a database, hosting, and authentication. Firebase's authentication service ensures secure user authentication and authorization. The project's significance lies in its potential applications across various domains of code explanation. The user-friendly interface, advanced NLP capabilities using LLM, and collaboration of different technologies enable the development of a powerful tool for explaining the code. Moreover, the project incorporates additional features such as the Save Option, allowing users to save generated explanations for future reference. The Saved Explanations Page Module provides a dedicated space for managing saved explanations, allowing users to view explanation as needed. Additionally, the Search Functionality enriches the user experience by providing a powerful tool to search saved explanations based on specific keywords or code snippets. In conclusion, the code explainer project represents a significant advancement in code comprehension technology, offering users a comprehensive solution for understanding code snippets. Through the seamless integration of various technologies and the incorporation of advanced features, the project aims to revolutionize the way code is explained and understood, catering to the needs of developers, students, and non-technical stakeholders alike.

IX. REFERENCE

[1] Exploring Large Language Models for Code Explanation (2023) Paheli Bhattacharya, Manojit Chakraborty, Kartheek N S N Palepu, Vikas Pandey, Ishan Dindorkar, Rakesh Rajpurohit, Rishabh Gupta.

[2] A GPTutor: a ChatGPT-powered programming tool for code explanation (2023) Eason Chen, Ray Huang, Han-Shin Chen, Yuen-Hsien Tseng, Liang-Yi Li.

[3] Llama 2: Open Foundation and Fine-Tuned Chat Models (2023) Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn.

[4] A Survey of Large Language Models (2023) Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, Ji-Rong Wen.

[5] A Comprehensive Overview of Large Language Models (2023) Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, Ajmal Mian.