EXPLAINABLE AI

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

RISHU RAJ

12204901

KO305

Roll no. 55

CSE 225

ANDROID APP DEVELOPMENT

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

March 2024

TABLE OF CONTENTS

INTRODUCTION

In the realm of Artificial Intelligence, Explainable AI (XAI) represents a pivotal advancement. Unlike conventional AI systems that solely provide answers, Explainable AI delves deeper by not only furnishing responses but also offering transparency into its decision-making process. This innovative approach aims to enhance user trust and comprehension of AI-generated outputs.

Our project, Explainable AI, embodies this cutting-edge concept by leveraging the Gemini API in the backend. Much like renowned AI models such as ChatGPT and Gemini, Explainable AI operates as a generative AI model. However, what sets it apart is its unparalleled ability to elucidate the rationale behind its responses, providing users with a comprehensive understanding of the information provided.

By integrating Explainable AI into your Android application, you are not only harnessing the power of advanced AI but also ensuring that your users receive answers accompanied by detailed explanations and the sources from which the AI has gleaned its knowledge. This novel approach is poised to revolutionize the way users interact with AI, fostering a deeper level of understanding and trust in the technology we have kept the App Name ***NORTOX***.

Modules or Activity Explanation for Explainable AI

1.Splash Screen: The splash screen is the first screen that appears when the Explainable AI app is launched. It typically displays the app's logo or branding and provides a brief introduction to the app's purpose. The splash screen helps create a seamless transition into the app and gives users a glimpse of what to expect.

2. Greeting Message: Following the splash screen, the app displays a greeting message to welcome the user. This message can be personalized based on the time of day (e.g., "Good morning," "Good afternoon," "Good evening", “Hii, how can I help you today?”) and can also include a brief message introducing the app or offering tips on how to use it effectively.

These features add a touch of professionalism and user-friendliness to the app, enhancing the overall user experience.

3. User Interface (UI): The UI module encompasses the design and layout of the Android application. It includes user interaction components such as buttons, text fields, and displays. The UI is crucial for providing a seamless and intuitive user experience.

4. Toast Message: When the user submits a question, a toast message saying "Wait, generating..." is displayed. This message indicates to the user that the app is processing their request and generating a response.

5. Submit Button: The submit button allows the user to submit their question to the Explainable AI model for processing. Upon clicking this button, the app initiates the process of generating a response based on the user's input.

6. Question Display Box: This box displays the user's current question or query. It provides a visual indication of the question being processed by the AI model and allows users to review their input before submitting it.

7. Input Processing: This module handles user inputs, such as questions or queries submitted to the AI. It processes these inputs and prepares them for the AI model to generate a response.

8. AI Model Integration: The heart of the application lies in its integration with the Gemini API, which powers the Explainable AI model. This module manages the communication with the API, sending user inputs and receiving AI-generated responses.

9. Response Generation: Once the AI model processes the input, this module handles the generation of the response. It not only provides the answer to the user's query but also includes the sources from which the AI has learned to arrive at that answer.

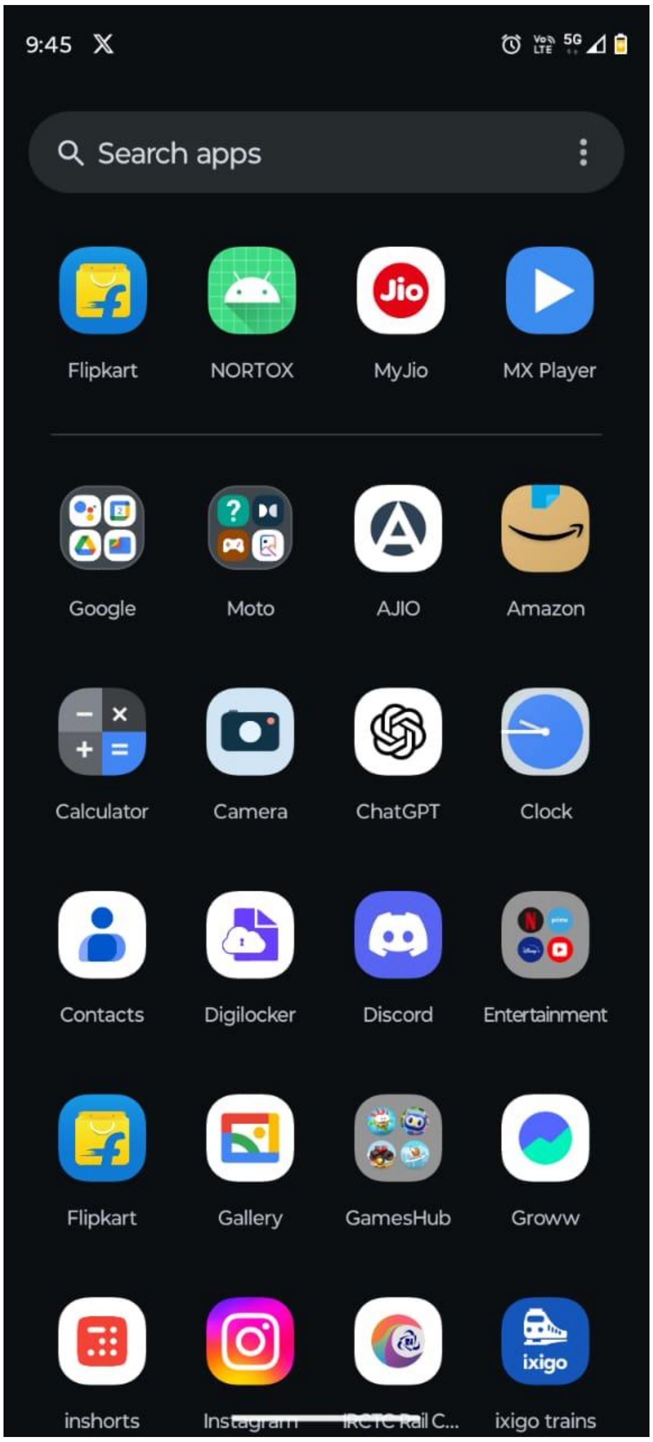
8. User Feedback and Improvement: This module gathers feedback from users regarding the accuracy and clarity of the AI's answers. This feedback is used to improve the AI model over time, enhancing its performance and explanatory capabilities.

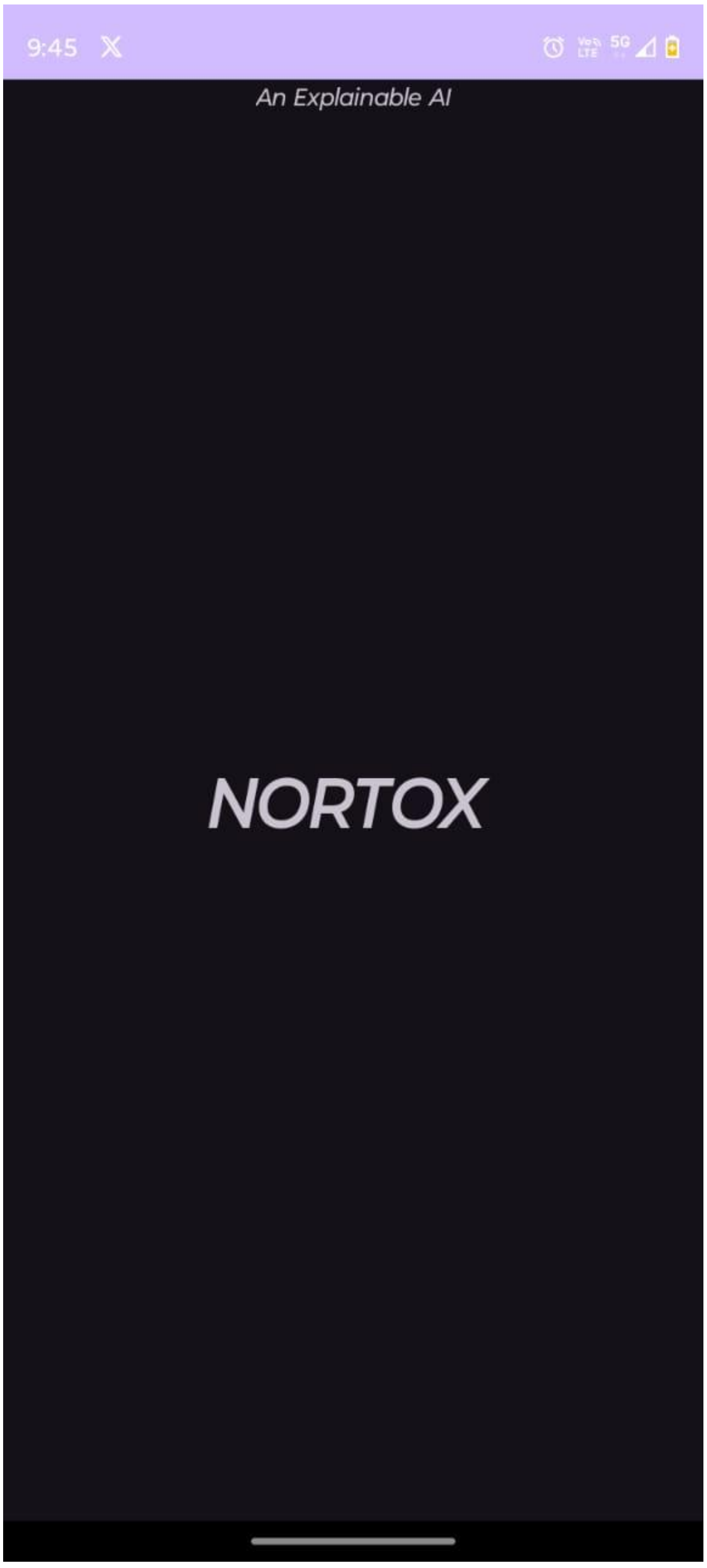
9. Error Handling and Reporting: To ensure the reliability of the application, this module manages errors that may occur during operation. It also provides mechanisms for users to report issues or inaccuracies in the AI's responses.

10. Data Privacy and Security: Given the sensitivity of the information processed by the application, this module ensures that user data is protected and that privacy regulations are adhered to.

These modules work in tandem to create a robust and user-friendly application that not only provides accurate answers but also offers transparent and informative explanations for those answers.

APP SCREENSHOTS:

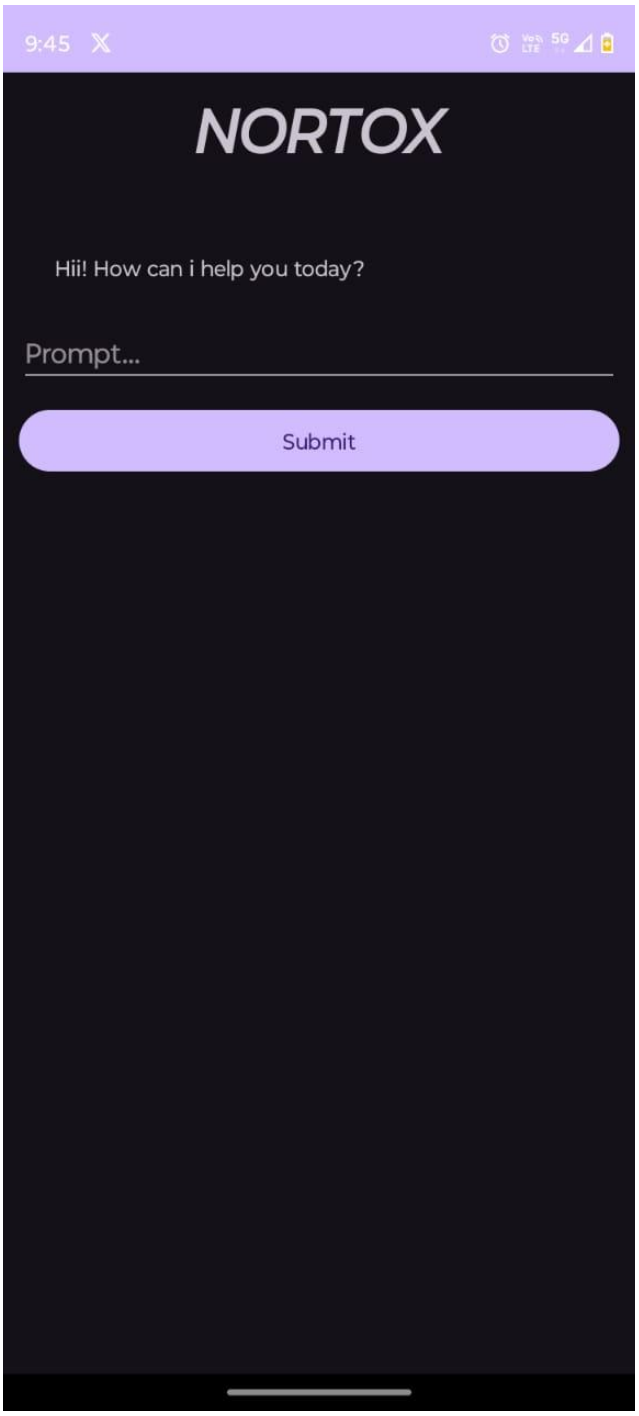


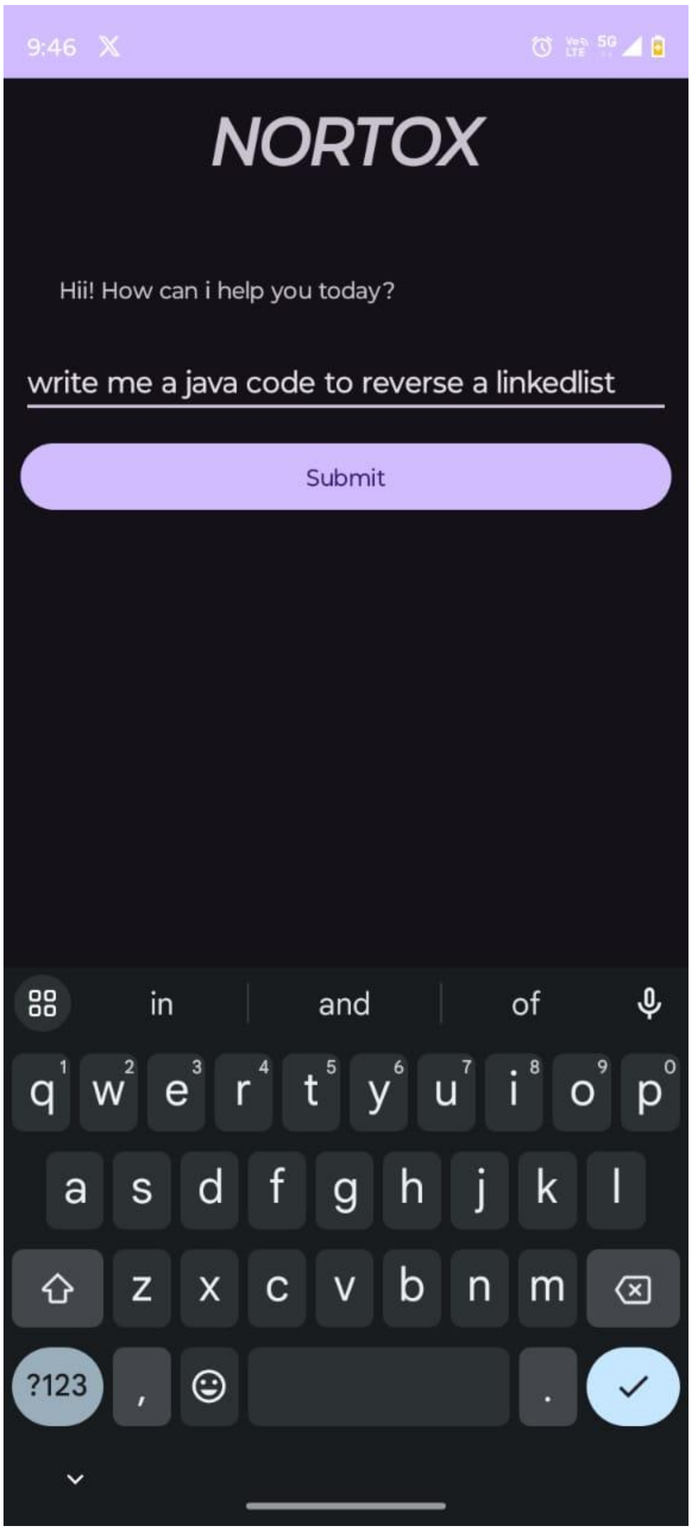


9:45

An Explainable AI

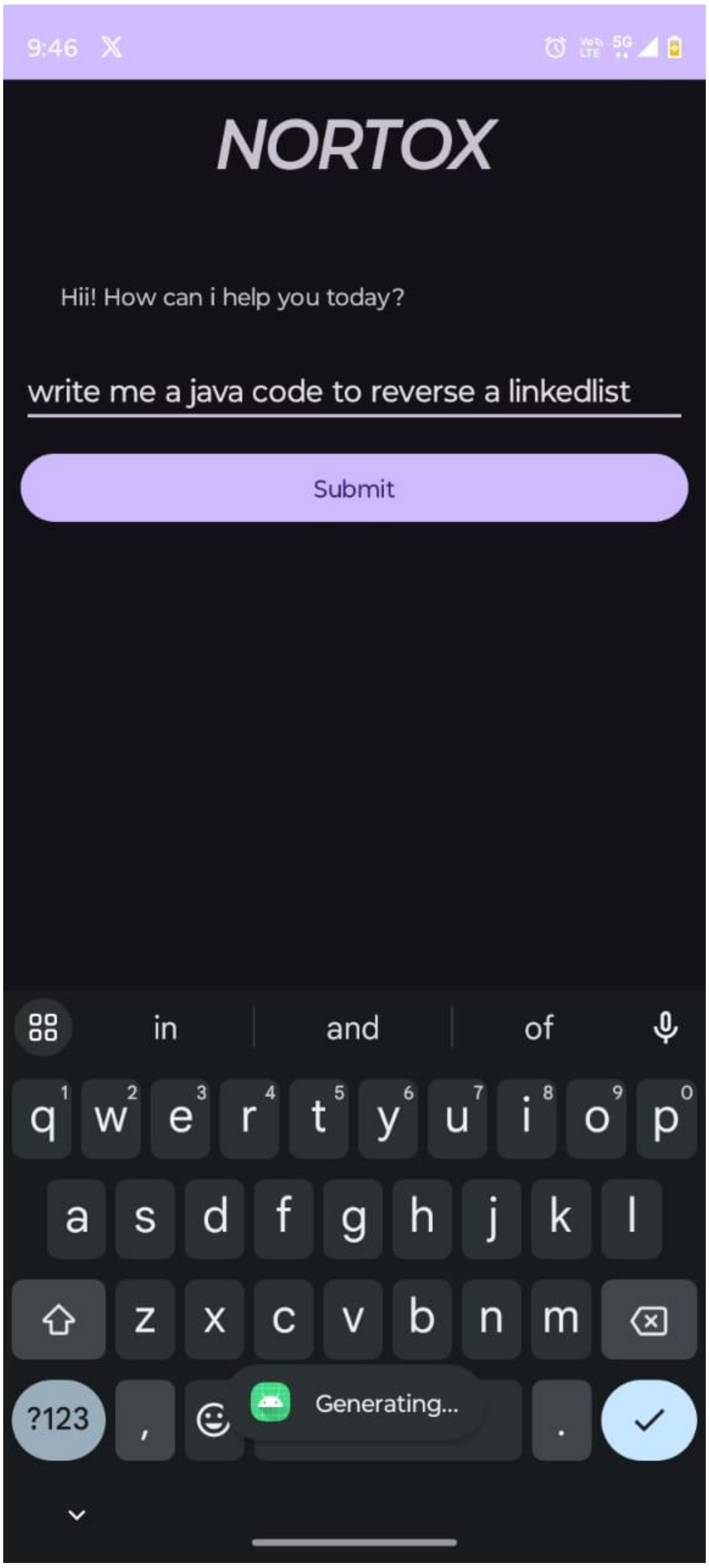
NORTOX


9:45
NORTOX
Hii! How can i help you today?
Prompt...
Submit


NORTOX

Hii! How can i help you today?

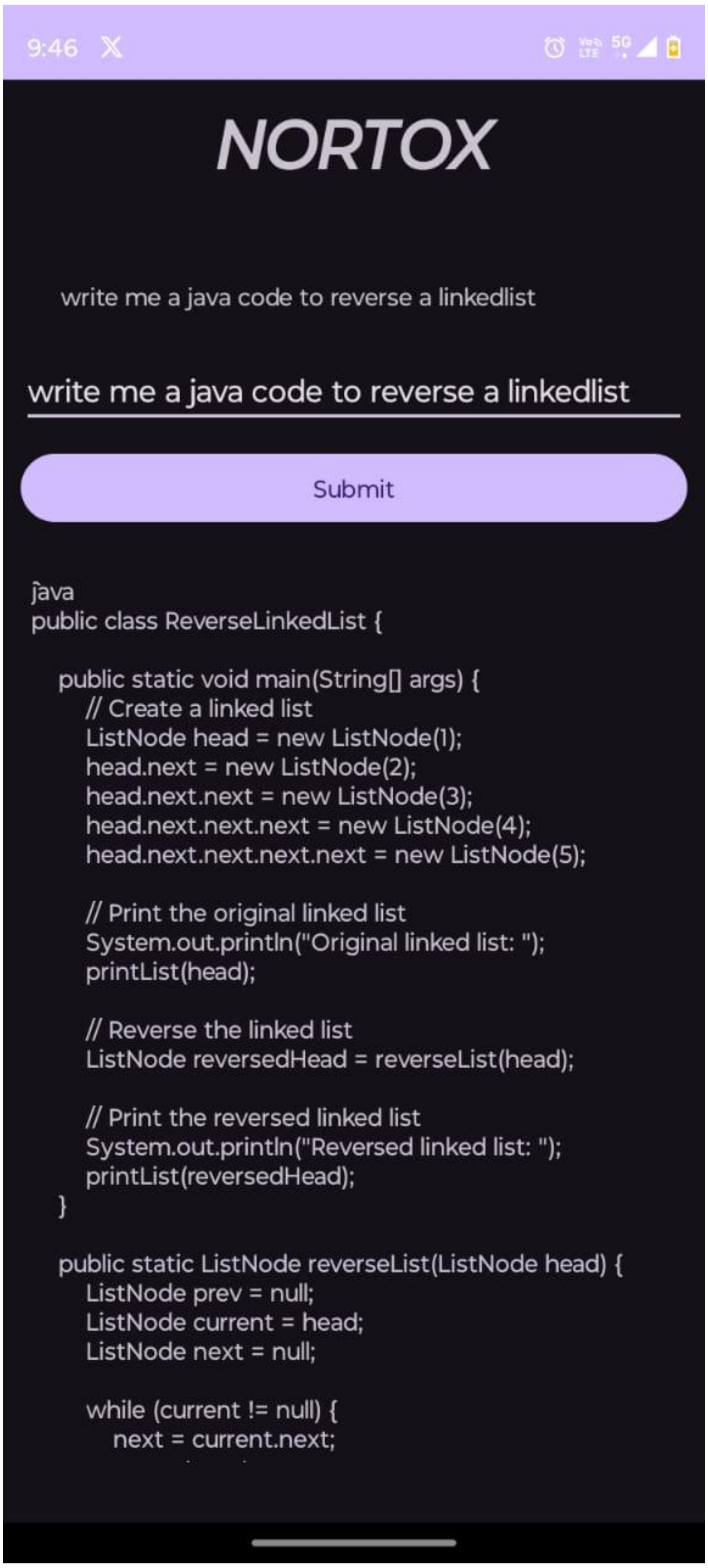
write me a java code to reverse a linkedlist


9:46

NORTOX

Hii! How can i help you today?

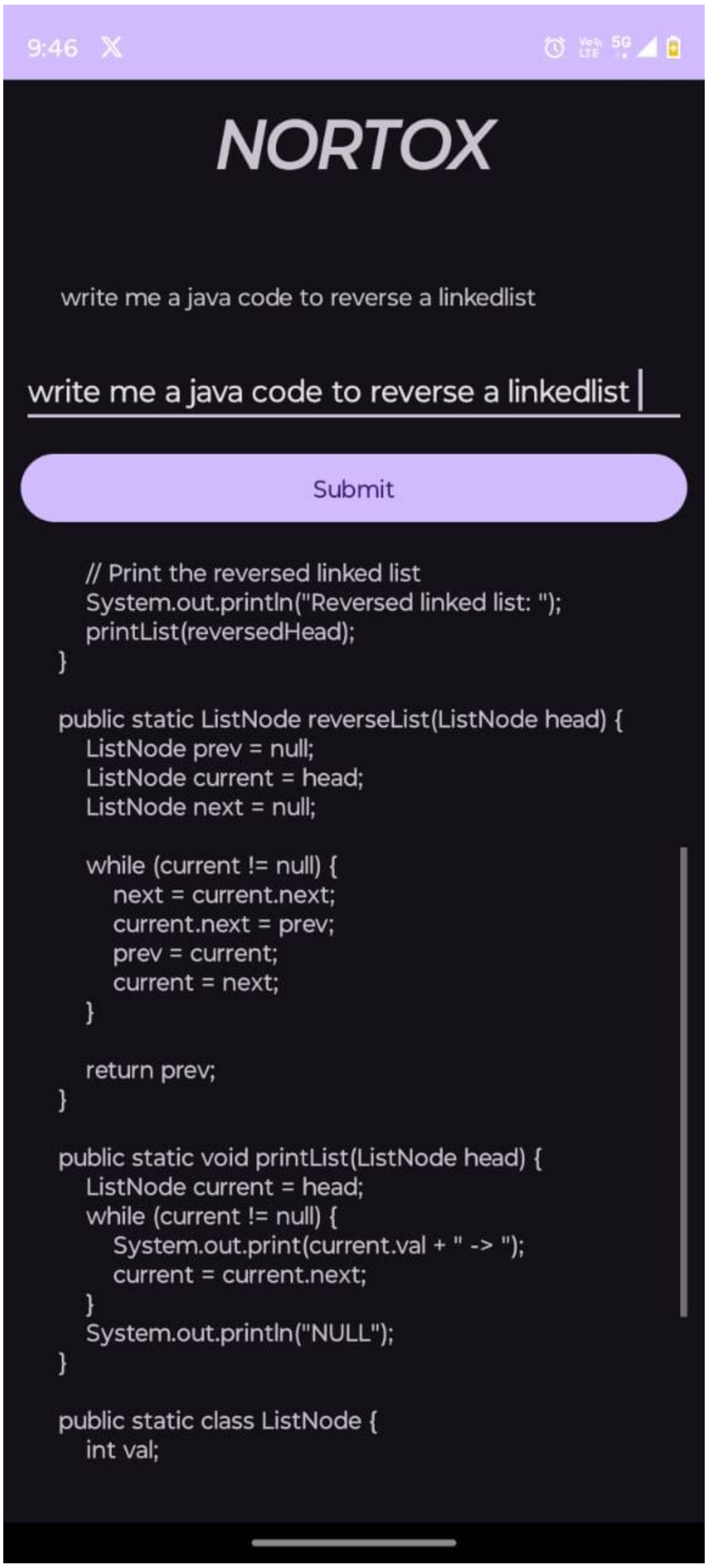
write me a java code to reverse a linkedlist



write me a java code to reverse a linkedlist

write me a java code to reverse a linkedlist

```
java
public class ReverseLinkedList {

    public static void main(String[] args) {
        // Create a linked list
        ListNode head = new ListNode(1);
        head.next = new ListNode(2);
        head.next.next = new ListNode(3);
        head.next.next.next = new ListNode(4);
        head.next.next.next.next = new ListNode(5);

        // Print the original linked list
        System.out.println("Original linked list: ");
        printList(head);

        // Reverse the linked list
        ListNode reversedHead = reverseList(head);

        // Print the reversed linked list
        System.out.println("Reversed linked list: ");
        printList(reversedHead);
    }

    public static ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode current = head;
        ListNode next = null;

        while (current != null) {
            next = current.next;
```

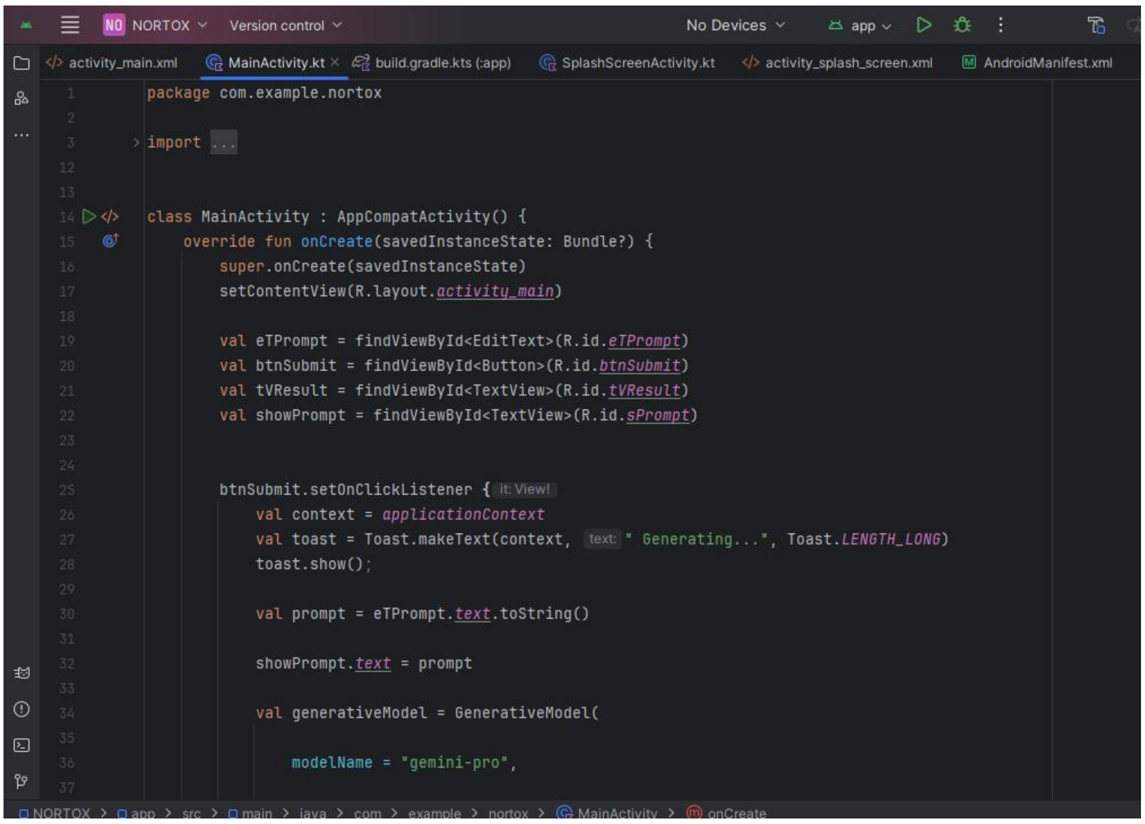


write me a java code to reverse a linkedlist

write me a java code to reverse a linkedlist

```
    // Print the reversed linked list
    System.out.println("Reversed linked list: ");
    printList(reversedHead);
}

public static ListNode reverseList(ListNode head) {
    ListNode prev = null;
    ListNode current = head;
    ListNode next = null;

    while (current != null) {
        next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }

    return prev;
}

public static void printList(ListNode head) {
    ListNode current = head;
    while (current != null) {
        System.out.print(current.val + " -> ");
        current = current.next;
    }
    System.out.println("NULL");
}

public static class ListNode {
    int val;
```

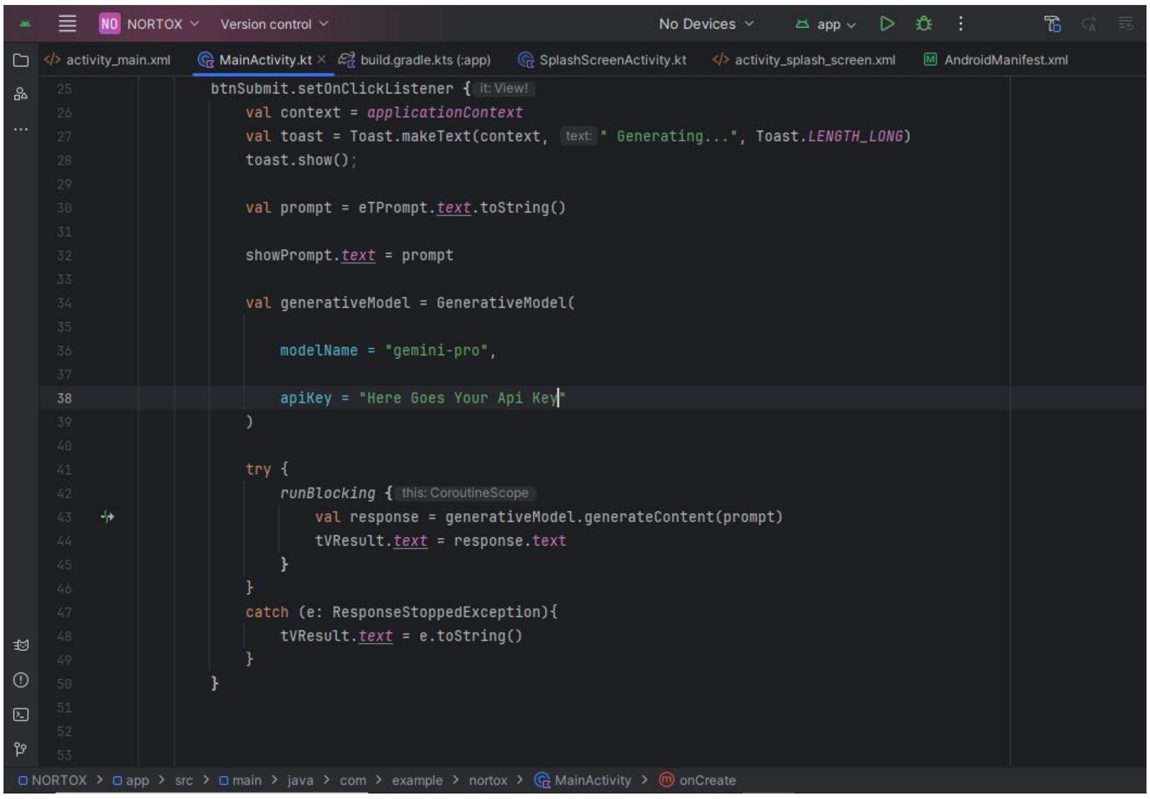
CODE SCREENSHOT:


NO NORTOX Version control No Devices app

activity_main.xml MainActivity.kt build.gradle.kts (:app) SplashScreenActivity.kt activity_splash_screen.xml AndroidManifest.xml

```kotlin
package com.example.nortox

import ...


class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val eTPrompt = findViewById<EditText>(R.id.eTPrompt)
        val btnSubmit = findViewById<Button>(R.id.btnSubmit)
        val tVResult = findViewById<TextView>(R.id.tVResult)
        val showPrompt = findViewById<TextView>(R.id.sPrompt)


        btnSubmit.setOnClickListener { it: View!
            val context = applicationContext
            val toast = Toast.makeText(context, text: " Generating...", Toast.LENGTH_LONG)
            toast.show();

            val prompt = eTPrompt.text.toString()

            showPrompt.text = prompt

            val generativeModel = GenerativeModel(

                modelName = "gemini-pro",

```

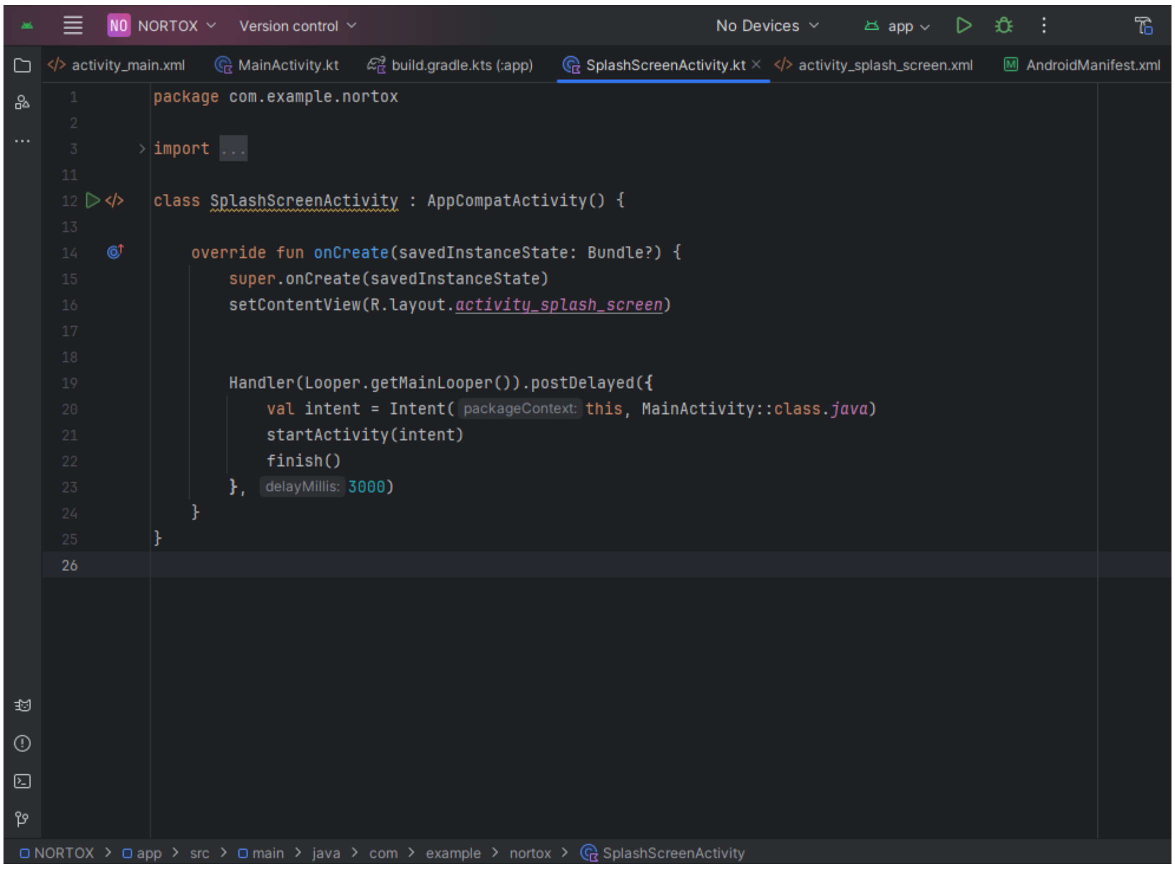
NORTOX > app > src > main > java > com > example > nortox > MainActivity > onCreate


NO NORTOX Version control No Devices app

activity_main.xml MainActivity.kt build.gradle.kts (:app) SplashScreenActivity.kt activity_splash_screen.xml AndroidManifest.xml

```kotlin
        btnSubmit.setOnClickListener { it: View!
            val context = applicationContext
            val toast = Toast.makeText(context, text: " Generating...", Toast.LENGTH_LONG)
            toast.show();

            val prompt = eTPrompt.text.toString()

            showPrompt.text = prompt

            val generativeModel = GenerativeModel(

                modelName = "gemini-pro",

                apiKey = "Here Goes Your Api Key"
            )

            try {
                runBlocking { this: CoroutineScope
                    val response = generativeModel.generateContent(prompt)
                    tVResult.text = response.text
                }
            }
            catch (e: ResponseStoppedException){
                tVResult.text = e.toString()
            }
        }



```

NORTOX > app > src > main > java > com > example > nortox > MainActivity > onCreate



```kotlin
package com.example.nortox

import ...

class SplashScreenActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash_screen)


        Handler(Looper.getMainLooper()).postDelayed({
            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
            finish()
        }, delayMillis: 3000)
    }
}
```

Conclusion

In conclusion, Explainable AI represents a significant advancement in AI technology, offering users not only answers to their queries but also detailed explanations and sources of its learning. Through the integration of the Gemini API, our Android application provides a transparent and informative AI experience, setting it apart from traditional AI models.

The implementation of features such as a splash screen, greeting message, toast messages, submit button, and question display box enhances the usability and functionality of the app, ensuring a seamless user experience. By focusing on clarity and transparency, Explainable AI aims to build trust and understanding between users and AI technology.

Future Scope

Looking ahead, there are several avenues for further enhancement and expansion of the Explainable AI app. One potential area of improvement is the integration of natural language processing (NLP) techniques to enhance the AI's ability to understand and respond to user queries more accurately.

Additionally, the app could be extended to support more languages and provide explanations in multiple languages, catering to a broader user base. Improvements in the explanation engine could also be explored to provide more detailed and contextually relevant explanations.

Furthermore, incorporating machine learning algorithms to analyze user feedback and improve the AI model's performance over time could enhance the app's effectiveness. Overall, the future scope of Explainable AI is promising, with ample opportunities for growth and innovation in the field of AI-driven applications.

GitHub Link: https://github.com/RishuRaj17/NORTOX-AI