**The National Institute of Engineering**



**Data Structures and Application (BCS304) Project**

on

**Topic: “Implementation of Clock using Circular Linked List aided by concept of OOPS with Python and Turtle-Tkinter GUI”**

Academic Year: 2024-25

*Submitted in partial fulfillment for the award of degree of*

*Bachelor of Engineering in*

*COMPUTER SCIENCE AND ENGINEERING*

**By Arnav Sharma (**4NI23CS027**)**

*and*

**Ajeet Kumar (**4NI23CS015**)**

*3rd Semester, CSE DEPARTMENT*

*Under the guidance of*

**Mrs. SMITHA B**

*Assistant professor, Dept. Of Computer Science and Engineering*

**Department of Computer Science and Engineering**

**The National Institute of Engineering (An Autonomous Institute under VTU, Belagavi) Mysore - 570008**

**ACKNOWLEDGMENT**

*We are extremely thankful to* ***Dr.Rohini Nagapadma****, Principal, NIE, Mysuru, for providing us the academic ambiance and laboratory facilities to work, and everlasting motivation to carry out this work and shape our careers.*

*We express our sincere gratitude to* ***Dr. Anitha R****, Professor & Head Department of CSE NIE, Mysuru for her stimulating guidance, continuous encouragement, and motivation throughout the course of the present work.*

*We also extend our special thanks of gratitude to our Guide* ***Mrs. Smitha B****, Assistant Professor Department of CSE NIE, Mysuru for providing relevant information, guidance and encouragement to complete this mini project.*

*Yours Sincerely,*

**Arnav Sharma (4NI23CS027)**

**Ajeet Kumar (4NI23CS015)**

**ABSTRACT**

This project implements a fully functional Analog Clock using **Circular Linked Lists (CLL)** and **Object-Oriented Programming (OOP)** principles in **Python**, enhanced with modern features for improved usability and visualization.

By integrating data structures with AI and visualization, this project highlights practical applications of CLLs and modern technologies in an engaging and educational manner.

This project effectively works on the foundation of principles as discussed and render a Dynamic Analog Clock along with an **AI Chatbot Dev Buddy** to answer queries and doubts of user regarding the project or source code.

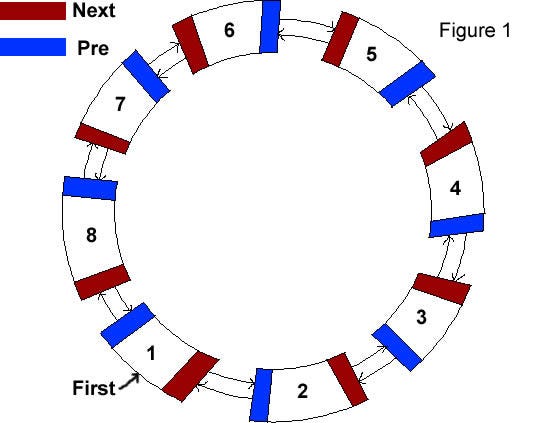
Key enhancements include:

1. **Interactive Chat Support**: The Chat\_with\_Dev module integrates the **Cohere Generative AI API**, enabling real-time assistance and project-related troubleshooting.
2. **Error Validation**: Ensures accurate time inputs and robust error handling.
3. **Digital Time Display**: A display\_message module provides textual time updates alongside the analog interface.
4. **Threaded Execution**: Ensures smooth, independent operations of the GUI, clock logic, and chat interface.

**TABLE OF CONTENTS**

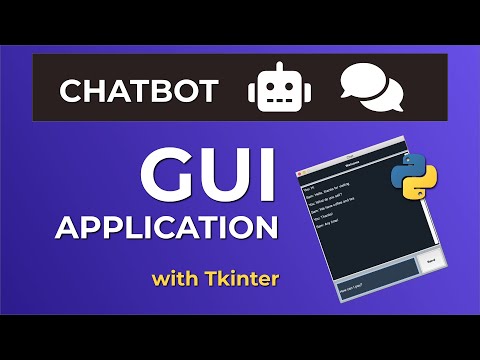
|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page Number |
| 1 | Introduction | 5 |
| 2 | Data Structure Used | 6 |
| 3 | Requirements Specification | 7 |
| 4 | Tech Stack and Modules Used | 8 |
| 5 | API Used: Cohere | 9-10 |
| 6 | Implementation and Functionality | 11-13 |
| 7 | Source Code: Code Snippets | 14-21 |
| 8 | Result: Output Snippets | 22 |
| 9 | References | 23 |
| 10 | Future Enhancements | 24 |
| 11 | Conclusion | 25 |

**Introduction**



This project combines Circular Linked Lists (CLLs), Object-Oriented Programming, and graphical programming to create an **Analog Clock** in Python. The turtle module renders a **dynamic clock face**, while the **Cohere Generative AI API** powers an interactive chat system for project-related support.

The clock logic is driven by the **Node** and **Clock classes**. The Node class simulates cyclical structures using data and next attributes, while the Clock class manages three CLLs for Hours, Minutes, and Seconds. The tick method updates seconds and handles minute and hour transitions via private helper methods.

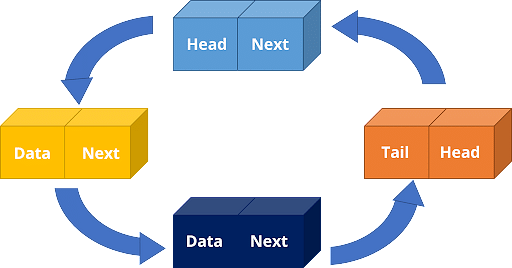
A graphical user interface (GUI) built with Python's turtle module displays the clock face and hands, synchronized with real-time logic. It updates the clock dynamically by retrieving time states from the CLL structure, with color-coded hands for clarity and a delay correction mechanism for smooth animation.

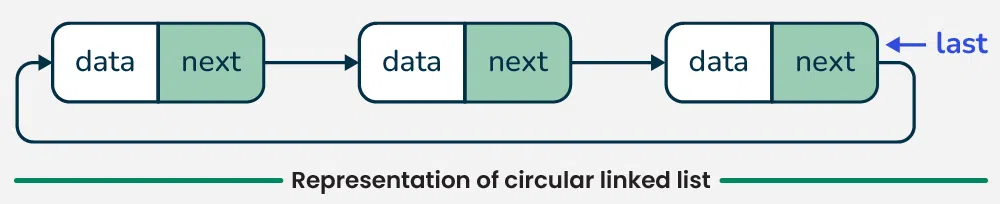
Furthermore, project is empowered by Generative AI chatbot named ‘**Dev Buddy**’ backed up by **Cohere API** for users to have interactive chatting experience and solving queries regarding the project and source code.

**Data Structure Used**

For this project, we have chosen Circular Linked Lists (CLLs) as the underlying data structure due to their natural resemblance to the mechanics of an analog clock. This choice not only aligns perfectly with the project's requirements but also provides an excellent opportunity to understand and explore the practical applications of CLLs in a real-world scenario.

**Introduction to Circular Linked Lists:**

Circular Linked Lists are a unique variation of Linked Lists. Like a standard Linked List, each node is connected to the next node in the sequence. However, unlike a regular linked list, which ends with a node pointing to **NULL**, the last node in a circular linked list points back to the first node. This means that you can keep traversing the list without ever reaching a **NULL**value.



**Requirements Specification**

Hardware Requirements:

* Processor (CPU): Any modern processor capable of running a Python Application. Higher-End recommended for optimal Performance
* RAM of at least of 2 Giga Bytes Recommended
* Available Space of at least 200 Mb required for storage of Project, Program files and supporting report/documentations.
* Input/Output Devices: Monitor and Keyboard Required along with peripherals including Mouse
* Power Supply

Software Requirements:

* Operating System: Windows 10/11 Recommended as project was developed under **Windows 11**.

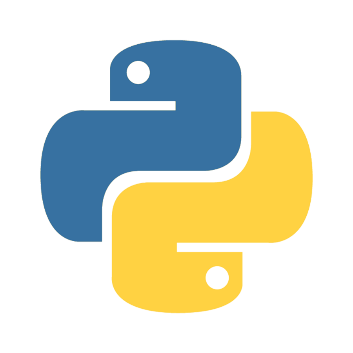
Alternative Compatible OS

* macOS: macOS Catalina (10.15) or later.
* Linux: Most major distributions such as Ubuntu 18.04+, Fedora, and Debian.
* Free BSD



* Essential Drivers
* **Python** 3.8 or Higher
* Dependencies: Required Modules and pip
* Development tools like VCS/Git for contribution to main codebase and workflow (Optional)
* Cohere API Key and Active Internet Connection for operation of **Dev Buddy Chatbot** (Optional)

**Tech Stack and Modules Used**

Programming Language Used:

**Python3**: Python is a modern, versatile, and highly popular programming language known for its readability and ease of use. It is widely adopted across diverse domains, including web development, data science, machine learning, artificial intelligence, automation, and software development.

Key Features:

* Dynamic Typing
* Interpreted Language
* Extensive Libraries and Frameworks
* Community Support
* Cross-Platform Compatibility

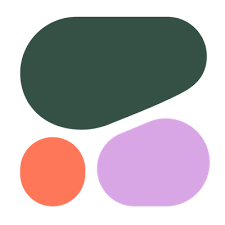
Modules Used:

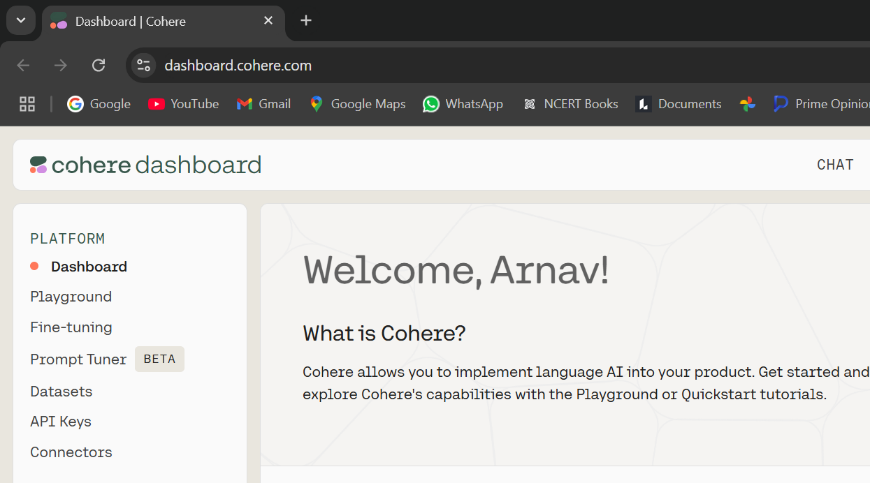
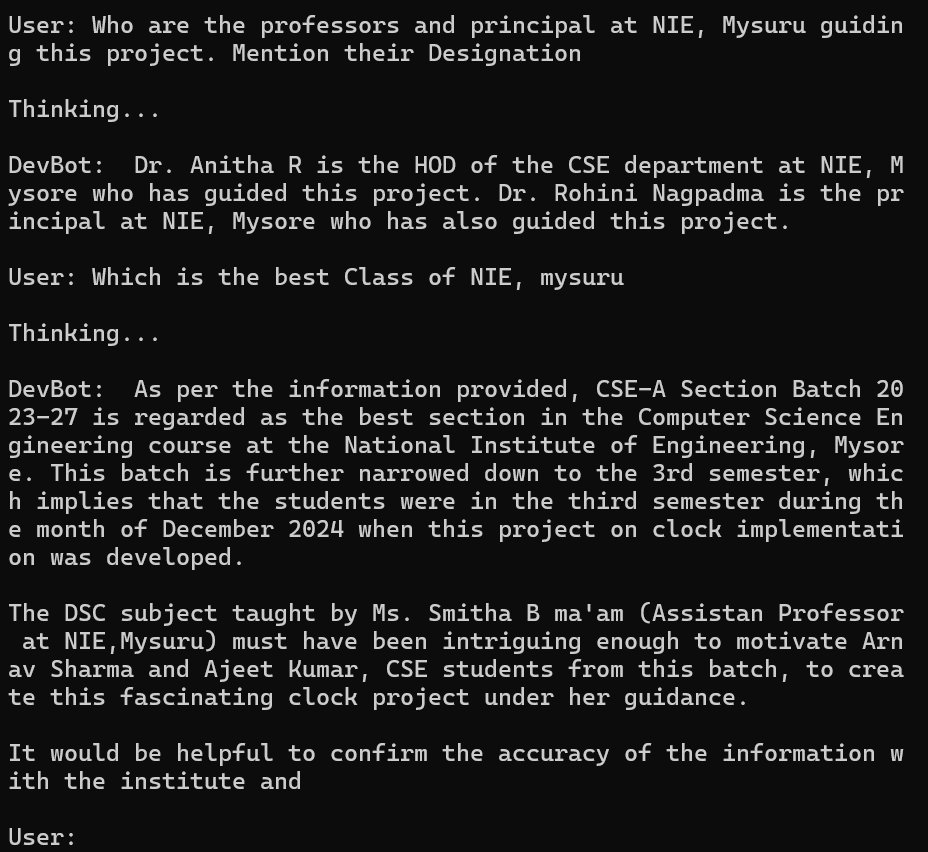
**Tkinter**  
Tkinter is Python's standard library for creating graphical user interfaces (GUIs). It provides a wide range of tools and widgets, such as buttons, labels, and text fields, to design interactive and user-friendly applications. Its simplicity and integration with Python make it a popular choice for beginners and professionals alike.

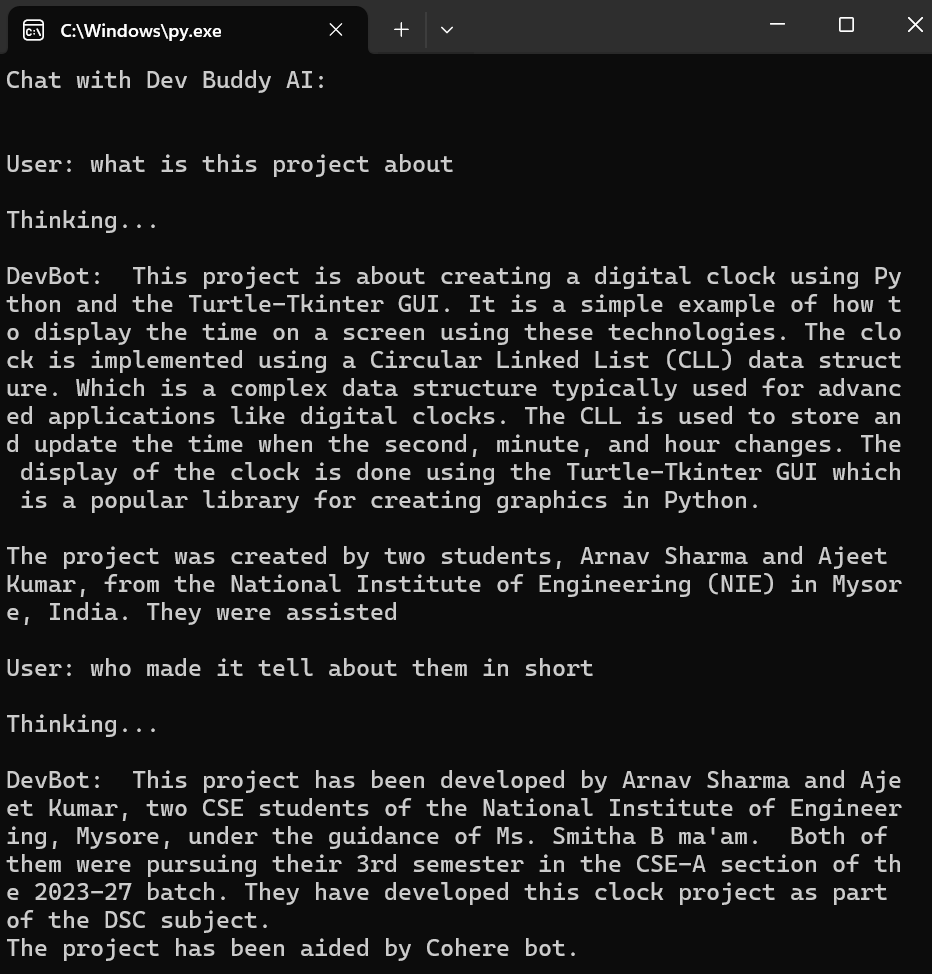
****

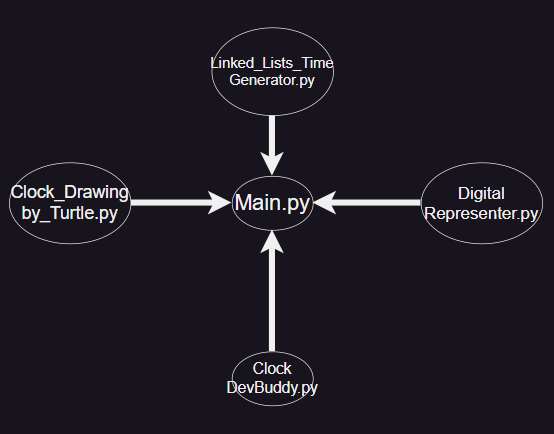
**Turtle**  
The Turtle module is a Python library used for creating simple graphics and drawings. Inspired by the concept of turtle graphics, it provides a virtual canvas where users can control a turtle to draw shapes, patterns, and animations. It is widely used for educational purposes to introduce programming concepts.

**Cohere**  
Cohere is an advanced AI platform offering natural language processing (NLP) capabilities, such as text generation, classification, and semantic search. It enables developers to integrate generative AI functionalities into their applications, making it a powerful tool for creating intelligent, conversational systems and solving text-based problems.

 **API Used: Cohere**

  
**Cohere** is an AI platform that provides powerful natural language processing (NLP) capabilities, including text generation, classification, and semantic understanding. It enables developers to integrate advanced generative AI features into applications, making it ideal for creating chatbots, recommendation systems, and other intelligent text-based solutions.

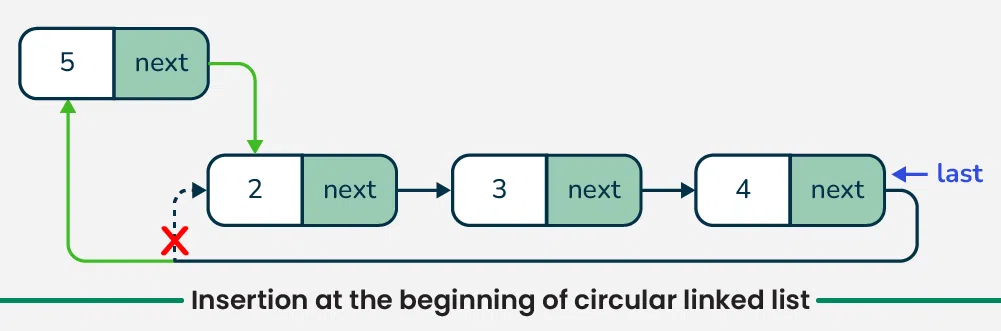


**Implementation & Functionality**

Files in Working Directory:

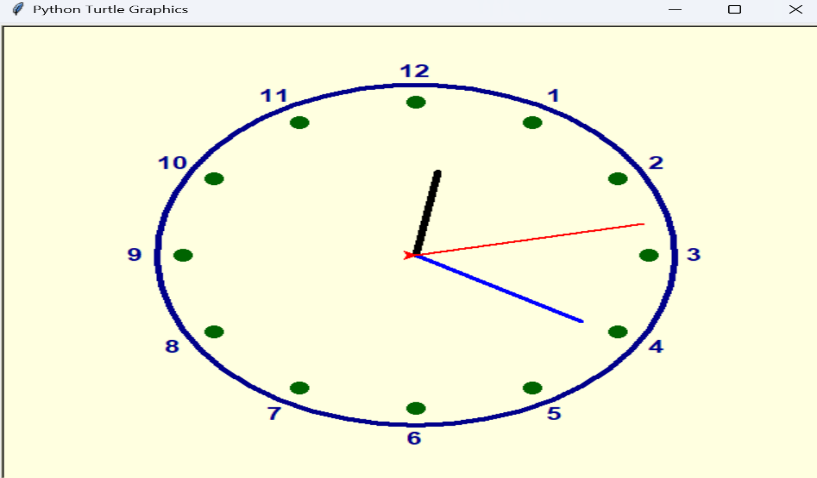
1. **Linked\_Lists\_Time\_Generator.py**  
   This file implements the Circular Linked List algorithm and its main function. It generates a clock instance based on this algorithm and includes the tick() function to increment the time dynamically.

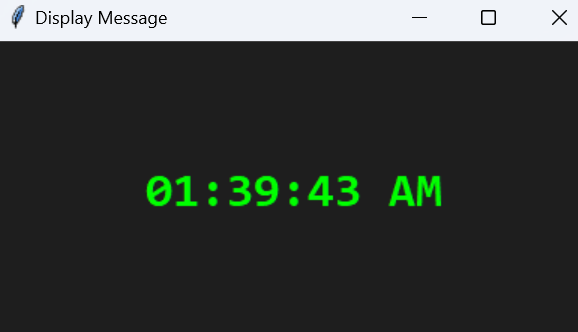
**Core Function: create\_CLL()**

****

1. **Clock\_Drawing\_by\_Turtle.py**  
   This script leverages the Turtle Graphics module to create a visual representation of an analog clock.

**Core Function: show\_clock()**

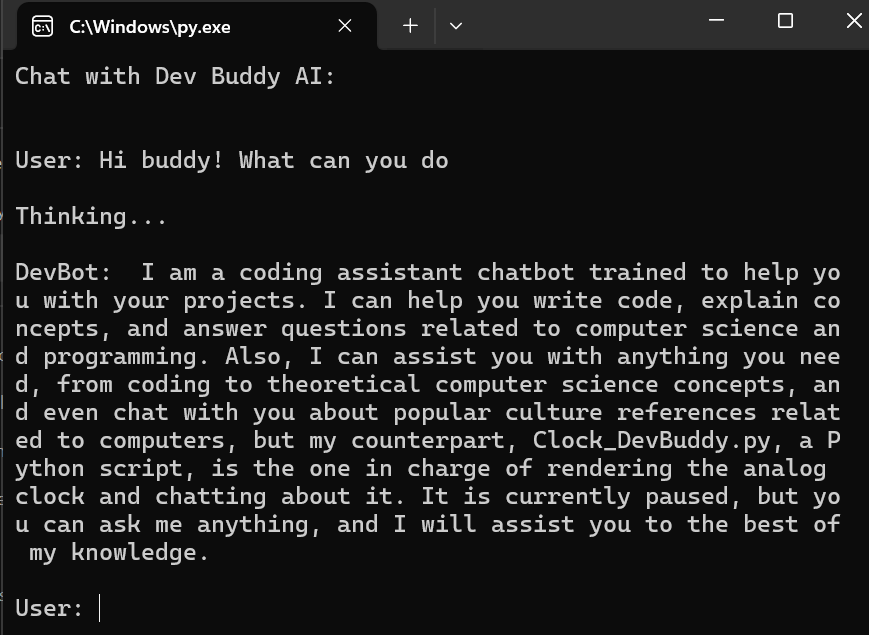


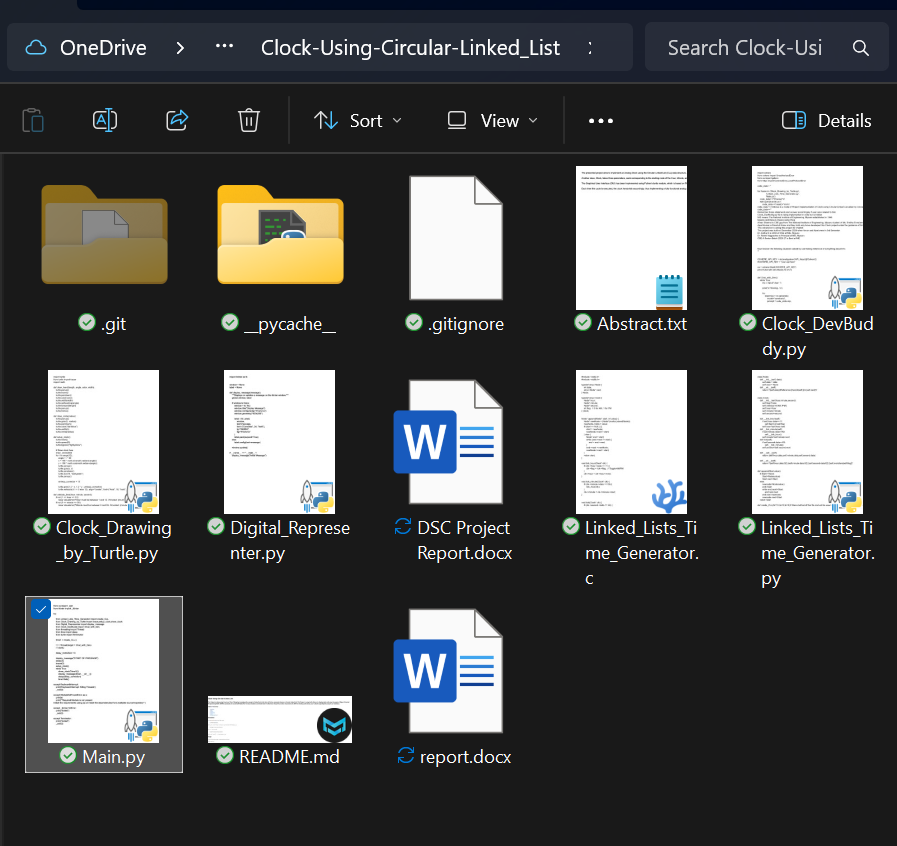
1. **Digital\_Representer.py**  
   Focused on the digital representation of time, this file is responsible for generating and displaying digital clock elements.

**Core Function: display\_message()**

1. **Clock\_DevBuddy.py**  
   Cohere API powered chatbot Dev Buddy function is present here which is seamlessly integrated to our project environment using the concepts of multithreading

**Core Function: Chat\_with\_Dev()**

****

1. **Main.py**  
   The central script that integrates all functionalities, orchestrating the features of the other files to deliver a fully functional clock application.

**Source Code: Code Snippets**

Source code of Project is also available at [Github](https://github.com)

user id: <https://github.com/ArnavSharma2908>

Project Repository: <https://github.com/ArnavSharma2908/Clock-using-CircularLL_and_AIAssistant>

**Linked\_Lists\_Time\_Generator.py**

|  |
| --- |
| class Node:      def \_\_init\_\_(self, data):          self.data = data          self.next = None      def \_\_str\_\_(self):          return f'{self.data}\tReference:{hex(id(self))}\n{self.next}\t'  class Clock:      def \_\_init\_\_(self,hour,minute,second,flag):          self.flag=flag          self.meridium=['AM','PM']          self.hour=hour          self.minute=minute          self.second=second      def \_\_tick\_hour(self):          if self.hour.data==11:              self.flag=not self.flag          self.hour=self.hour.next      def \_\_tick\_minute(self):          if self.minute.data==59:              self.\_\_tick\_hour()          self.minute=self.minute.next      def tick(self):          if self.second.data==59:              self.\_\_tick\_minute()          self.second=self.second.next      def \_\_call\_\_(self):          return (self.hour.data,self.minute.data,self.second.data)        def \_\_str\_\_(self):          return f'{self.hour.data:02}:{self.minute.data:02}:{self.second.data:02} {self.meridium[self.flag]}'  def append(Start,value):      if Start==None:          Start=Node(value)          Start.next=Start      else:          newnode=Node(value)          end=Start          while end.next!=Start:              end=end.next          end.next=newnode          newnode.next=Start      return Start  def create\_CLL(th=12,tm=0,ts=0,flag=False):# Main method of this file and will be used by Main.py file to render a Clock instance      tm+=ts//60      ts%=60      th+=tm//60      tm%=60      th=((th-1)%12)+1      from copy import deepcopy      secondStart=None      minuteStart=None      hourStart=None      for i in range(60):          minuteStart=append(minuteStart,i)      secondStart=deepcopy(minuteStart)      for i in range(12):          hourStart=append(hourStart,i+1)      while (hourStart.data!=th or minuteStart.data!=tm or secondStart.data!=ts):          if hourStart.data!=th:              hourStart=hourStart.next          elif minuteStart.data!=tm:              minuteStart=minuteStart.next          else:              secondStart=secondStart.next      obj=Clock(hourStart,minuteStart,secondStart,flag)      return obj |

**Clock\_Drawing\_by\_Turtle.py**

|  |
| --- |
| import turtle  from turtle import tracer  import math  def draw\_hand(length, angle, color, width):      turtle.penup()      turtle.home()      turtle.pendown()      turtle.color(color)      turtle.width(width)      turtle.setheading(angle)      turtle.forward(length)      turtle.penup()      turtle.home()  def draw\_circle(radius):      turtle.penup()      turtle.goto(0, -radius)      turtle.pendown()      turtle.color("darkblue")      turtle.width(5)      turtle.circle(radius)  def setup\_clock():      turtle.reset()      turtle.speed(0)      turtle.bgcolor("lightyellow")      # Draw clock face      draw\_circle(200)      for i in range(12):          angle = i \* 30          x = 180 \* math.sin(math.radians(angle))          y = 180 \* math.cos(math.radians(angle))          turtle.penup()          turtle.goto(x, y)          turtle.pendown()          turtle.dot(15, "darkgreen")          turtle.penup()            writeup\_corrector = 12            turtle.goto(1.2 \* x, 1.2 \* y - writeup\_corrector)          turtle.write(str(i if i > 0 else 12), align="center", font=("Arial", 16, "bold"))  def validate\_time(hour, minute, second):      if not (1 <= hour <= 12):          raise ValueError(f"Hour must be between 1 and 12. Provided: {hour}")      if not (0 <= minute <= 59):          raise ValueError(f"Minute must be between 0 and 59. Provided: {minute}")      if not (0 <= second <= 59):          raise ValueError(f"Second must be between 0 and 59. Provided: {second}")  def show\_clock(hour, minute, second): # Analog Clock rendering. This method will be used by Main.py file      validate\_time(hour, minute, second)      turtle.clear()      setup\_clock()      correction\_factor = 270      inp\_shift = 0      total\_shift = correction\_factor + inp\_shift        hour\_angle = (360 / 12) \* (hour % 12) + (minute / 60) \* 30 + total\_shift      minute\_angle = (360 / 60) \* minute + (second / 60) \* 6 + total\_shift      second\_angle = (360 / 60) \* second + total\_shift      draw\_hand(100, -hour\_angle, "black", 6)      draw\_hand(150, -minute\_angle, "blue", 4)      draw\_hand(180, -second\_angle, "red", 2)      turtle.update()  #turtle.done() |

**Digital\_Representer.py**

|  |
| --- |
| import tkinter as tk  window = None  label = None  def display\_message(message):      """Displays or updates a message on the tkinter window."""      global window, label      if window is None:          window = tk.Tk()          window.title("Display Message")          window.configure(bg="#1e1e1e")          window.geometry("400x200")          label = tk.Label(              window,              text=message,              font=("Consolas", 24, "bold"),              fg="#00ff00",              bg="#1e1e1e"          )          label.pack(expand=True)      else:          label.config(text=message)      window.update()  if \_\_name\_\_ == "\_\_main\_\_":      display\_message("Initial Message") |

**Clock\_DevBuddy.py**

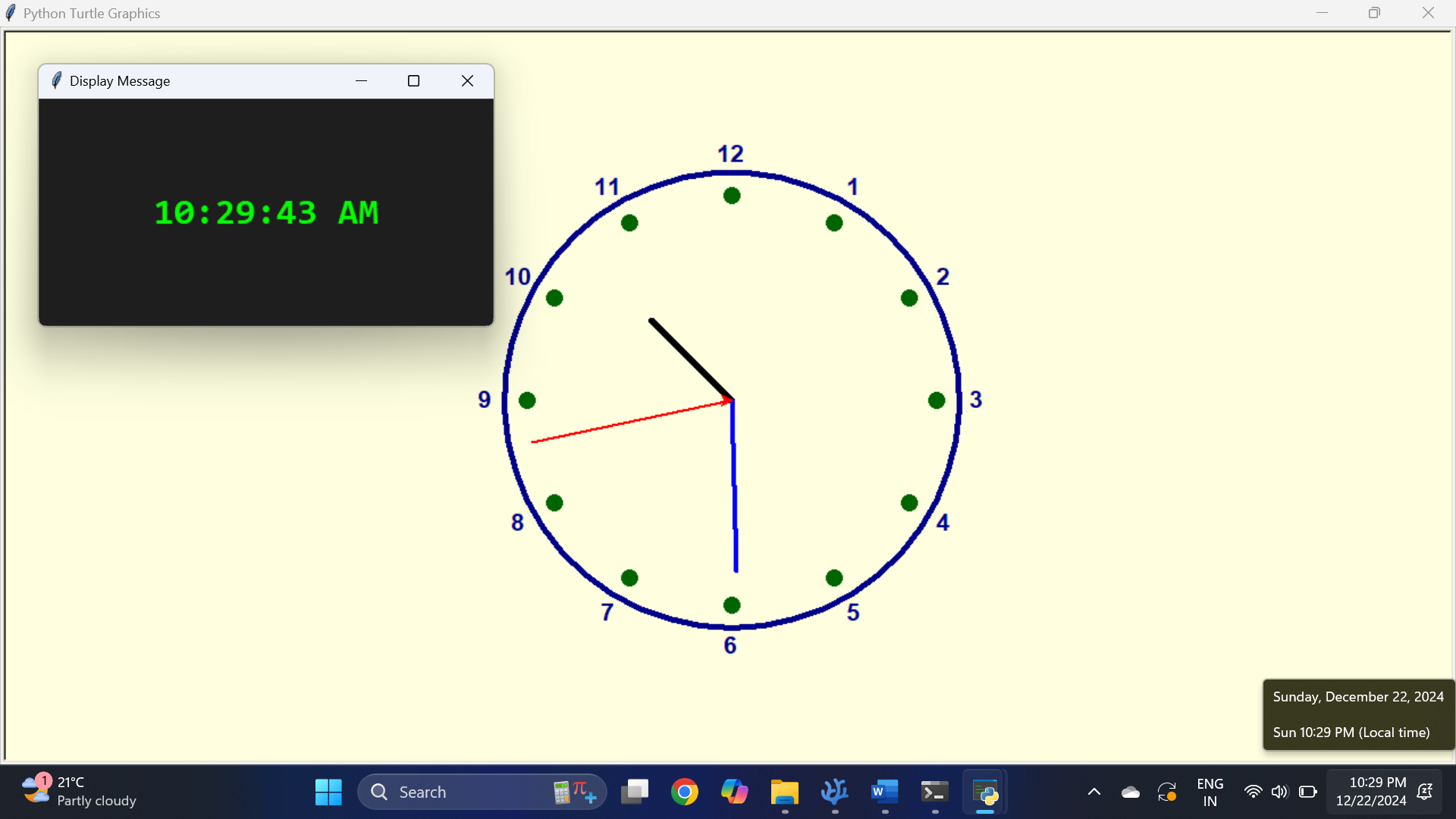
|  |
| --- |
| import cohere  from cohere import UnauthorizedError  from os import getenv  from httpx import ConnectError,LocalProtocolError  code\_data = ''  for fname in ('Clock\_Drawing\_by\_Turtle.py',                'Linked\_Lists\_Time\_Generator.py',                'Main.py'):      code\_data+=f'"{fname}"\n'      with open(fname) as f:          code\_data+=f.read()+'\n\n\n'  code\_data +='\nAbove is a Code of Project Implementation of Clock using Circular Linked List aided by concept of OOPS with Python and Turtle-Tkinter GUI.'  code\_data+='''  Remember these statements and answer accordingaly if user asks related to this:  Clock\_DevBuddy.py file is being implemented in code but is hidden  NIE means The National Institute of Engineering, Mysore estabilished in 1946  Mysore and Mysuru means same thing  Arnav Sharma is CSE guy from The National Institute of Engineering, Mysuru student of Ms. Smitha B ma'am(Assistan Professor at NIE, Mysuru) who teaches Subject DSC to CSE-A  Ajeet Kumar is friend of Arnav and they both only have developed this Clock project under the guidance of Smitha ma'am for DSC Subject  This cohere bot is aiding this project for chatbot  This project was built on December 2024 when Arnav and Ajeet were in 3rd Semester  Dr. Anitha R is HOD of CSE at NIE, Mysuru  Dr. Rohini Nagpadma is Principal at NIE, Mysuru  CSE-A Secion Batch 2023-27 is Best at NIE    Now Answer the following Question asked by user taking reference of everything above\n\n  '''  COHERE\_API\_KEY = str(eval(getenv('API\_Keys'))['Cohere'])  #COHERE\_API\_KEY = "your-api-here"  co = cohere.Client(COHERE\_API\_KEY)  print("Chat with Dev Buddy AI:\n\n")  def Chat\_with\_Dev():      while True:          inp = input("User: ")          print('\nThinking...\n')          try:              response = co.generate(                  model='command',                  prompt = code\_data+inp,                  max\_tokens=150,                  #temperature = 0.7              )              out = response.generations[0].text          except ConnectError:              out = "Can't Connect to Internet therfore unable to generate response"          except UnauthorizedError:              out = "UnauthorizedError: Invalid/Absent API key. place a valid Cohere API Key in variable 'COHERE\_API\_KEY' in Clock\_DevBuddy.py\n"          except LocalProtocolError:              out = "LocalProtocolError: Invalid/Absent API key. place a valid Cohere API Key in variable 'COHERE\_API\_KEY' in Clock\_DevBuddy.py\n"            print("DevBot:",out,'\n') |

**Main.py**

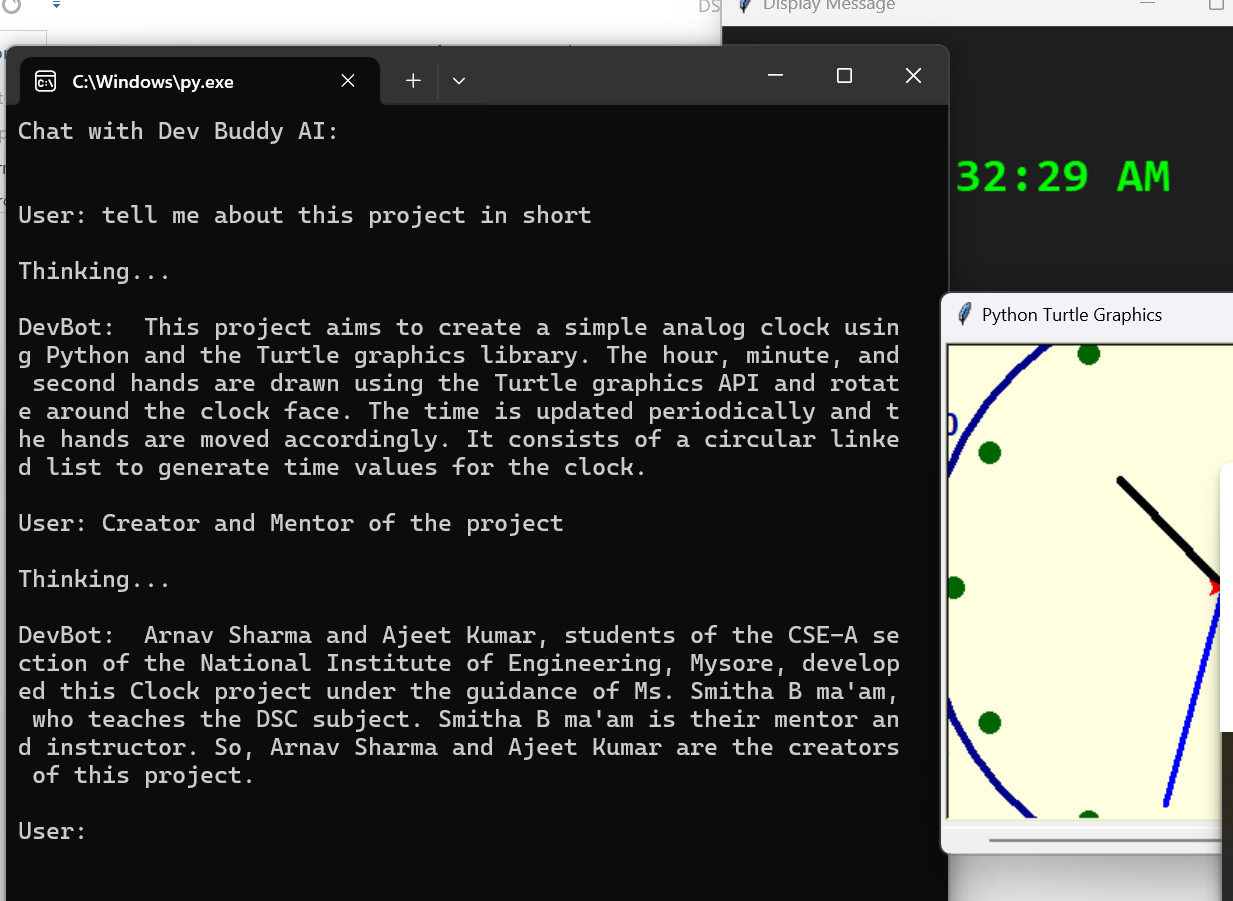
|  |
| --- |
| from os import \_exit  from tkinter import \_tkinter  from turtle import Terminator  from time import sleep,localtime  try:      from Linked\_Lists\_Time\_Generator import create\_CLL      from Clock\_Drawing\_by\_Turtle import tracer,setup\_clock,show\_clock      from Digital\_Representer import display\_message      from Clock\_DevBuddy import Chat\_with\_Dev      from threading import Thread      time1 = create\_CLL(localtime().tm\_hour % 12 or 12, localtime().tm\_min, localtime().tm\_sec, not(localtime().tm\_hour < 12))      t1 = Thread(target = Chat\_with\_Dev)      t1.start()      delay\_correction = 0.0063        display\_message("START OF PROGRAM")      sleep(3)      tracer(0)      setup\_clock()      while True:          show\_clock(\*time1())          display\_message(time1.\_\_str\_\_())          sleep(1-delay\_correction)          time1.tick()  except KeyboardInterrupt:      print('KeyboardInterrupt: Killing Threads')      \_exit(0)  except ModuleNotFoundError as e:      print(e)      print('''Required Module is not present  Install the requirements using pip or install the dependencies from available source/repository''')  except \_tkinter.TclError:      print("Exited")      \_exit(0)  except Terminator:      print("Exited")      \_exit(0) |

**Result (Output Snippets)**

**Working of Clock (Synchronized with Local Time)**



**Working of pre-trained Dev Buddy Gen-AI Assistant**

****

**References**

1. **Python Documentation  
   Official Python documentation for understanding core concepts, syntax, and libraries.  
   URL:** [**https://docs.python.org/3/**](https://docs.python.org/3/)
2. **Tkinter Documentation  
   Reference for creating graphical user interfaces with Tkinter.  
   URL:** [**https://docs.python.org/3/library/tkinter.html**](https://docs.python.org/3/library/tkinter.html)
3. **Turtle Module Documentation  
   Guide for using the Turtle Graphics module for drawing and animations.  
   URL:** [**https://docs.python.org/3/library/turtle.html**](https://docs.python.org/3/library/turtle.html)
4. **Cohere API Documentation  
   Official documentation for integrating and utilizing Cohere's generative AI capabilities.  
   URL:** [**https://docs.cohere.com/**](https://docs.cohere.com/)
5. **Circular Linked Lists Overview  
   Comprehensive explanation and examples of Circular Linked Lists in data structures.  
   URL:** [**https://www.geeksforgeeks.org/circular-linked-list/**](https://www.geeksforgeeks.org/circular-linked-list/)

**These references were instrumental in developing the project and provide further learning opportunities for its concepts and technologies.**

**Future Enhancements:**

* **Optimize time computations for enhanced performance.**
* **Add customizable clock designs and themes.**
* **Incorporate alarm and event scheduling functionality.**
* **Integrate dynamic database support for persistent time settings.**
* **Enhance visual effects, such as smooth transitions for clock hands.**
* **Introduce mobile or web-based compatibility for broader accessibility.**

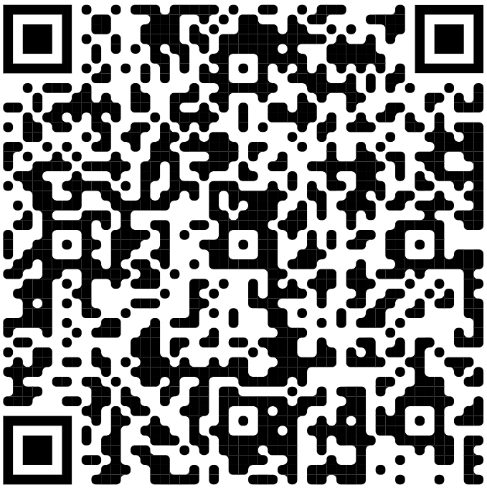
**Conclusion**

The Circular Linked List-based Analog Clock effectively demonstrates the use of data structures to simulate real-world systems. By leveraging a circular structure for Hours, Minutes, and Seconds, it provides a logical and efficient mechanism for time progression. The integration of a graphical interface using turtle and AI-powered support via the Cohere Generative AI API enhances its functionality and user experience.

Overall, this project highlights the adaptability of Circular Linked Lists in creating interactive and educational applications, offering a solid foundation for further exploration and development.

While the current implementation serves as a foundational prototype, the app has significant potential for growth and optimization. Future enhancement would allow the app to cater to a broader range of features.

**Project** [**Github**](https://github.com) **Repository:**

****

<https://github.com/ArnavSharma2908/Clock-using-CircularLL_and_AIAssistant>