Week 1 Project Report

This Week 1 project report is a detailed record of the project completed during this time. It includes important information such as the project's title, the technology used, the time spent, challenges faced, and a step-by-step explanation of how the project was developed. Additionally, this report outlines the assignments completed, including their titles, the specific tasks performed, and, if applicable, a summary of the topics covered in any presentations.

Project Details

1. Project Name:

Digital Clock GUI in Python

2. Technology Used:

- Programming Language: Python

- Libraries/Frameworks: Tkinter (for GUI development), time (for time manipulation)

3. Time Invested and Challenges:

- Time Invested: Approximately 15 hours over the week.

- Challenges:

  - Understanding the Tkinter library and its components.

  - Managing real-time updates in the GUI.

  - Ensuring the GUI remains responsive during updates.

  - Debugging issues related to the clock display and formatting.

4. Development Process:

Step 1: Project Conception

- Objective: To create a graphical user interface (GUI) that displays the current time, updating every second.

- Initial Research:Reviewed documentation and tutorials on Tkinter and its widgets to understand the basics of GUI development in Python.

Step 2: Setting Up the Development Environment

- Tools Installed: Installed Python, Tkinter library (included in standard Python distribution), and a code editor (such as Visual Studio Code or PyCharm).

Step 3: Basic GUI Layout

- Window Creation: Created a basic window using Tkinter.

```
  import tkinter as tk


  root = tk.Tk()

  root.title("Digital Clock")

  root.geometry("400x200")
```

- Label Widget: Added a label to display the time.

```
  time_label = tk.Label(root, font=('Helvetica', 48), bg='black', fg='white')

  time_label.pack(expand=True)
  ```
```

**Step 4: Displaying the Time**

- Time Function: Created a function to fetch the current time using the `time` library.

```
  import time
```
```

```
 def get_time():
     return time.strftime("%H:%M:%S")
```

Step 5: Updating the Time Display

- Update Function: Developed a function to update the label every second.

```
 def update_clock():
     current_time = get_time()
     time_label.config(text=current_time)
     root.after(1000, update_clock)
 ```
```

- Initial Call: Called the update function to start the clock.

```
 update_clock()
```

Step 6: Running the Application

- Main Loop: Ensured the Tkinter main loop keeps running to display the GUI.

```
 root.mainloop()
```

Step 7: Refinements and Debugging

- Layout Adjustments:Fine-tuned the window size, colors, and font styles for better aesthetics.

- Error Handling:Added basic error handling to manage unexpected issues.

Step 8: Testing and Validation

- Testing: Ran the application multiple times to ensure it consistently displayed the correct time and remained responsive.
```

Week 2 Project Report

This Week 2 project report is a detailed record of the project completed during this time. It includes important information such as the project's title, the technology used, the time spent, challenges faced, and a step-by-step explanation of how the project was developed. Additionally, this report outlines the assignments completed, including their titles, the specific tasks performed, and, if applicable, a summary of the topics covered in any presentations.

Project Details

1. Project Name:

URL Shortening in Python

2. Technology Used:

- **Programming Language**:Python
- **Libraries/Frameworks**: Tkinter (for GUI development), pyshorteners (for URL shortening), pyperclip (for clipboard management)

3. Time Invested and Challenges:

- Time Invested: Approximately 18 hours over the week.
- Challenges:
  - Understanding the integration of Tkinter with other libraries.
  - Managing user inputs and displaying outputs in a user-friendly manner.
  - Ensuring the application is responsive and handles errors gracefully.

4. Development Process:

Step 1: Project Conception

- Objective:To create a graphical user interface (GUI) application that allows users to shorten URLs and copy the shortened URL to the clipboard.

- Initial Research: Reviewed documentation and tutorials on Tkinter, pyshorteners, and pyperclip to understand their functionalities.

Step 2: Setting Up the Development Environment

- Tools Installed: Installed Python, Tkinter (included in standard Python distribution), pyshorteners, and pyperclip using pip.

```
pip install pyshorteners pyperclip
```

Step 3: Basic GUI Layout

- Window Creation:Created a basic window using Tkinter.

```
  import tkinter as tk


  root = tk.Tk()
  root.title("URL Shortener")
  root.geometry("400x200")
```

-Widgets: Added entry fields for input URL and buttons for actions.

```
 url_entry = tk.Entry(root, width=50)
 url_entry.pack(pady=10)

 shorten_button = tk.Button(root, text="Shorten URL", command=shorten_url)
 shorten_button.pack(pady=5)

 output_label = tk.Label(root, text="")
 output_label.pack(pady=10)
```

Step 4: URL Shortening Logic

- Shorten Function: Implemented a function to shorten the URL using `pyshorteners`.

~~~
 ```python
 import pyshorteners

 def shorten_url():
     s = pyshorteners.Shortener()
     long_url = url_entry.get()
     try:
         short_url = s.tinyurl.short(long_url)
         output_label.config(text=short_url)
         pyperclip.copy(short_url)
     except Exception as e:
         output_label.config(text="Error shortening URL")
~~~

Step 5: Clipboard Management

- Clipboard Function: Integrated `pyperclip` to copy the shortened URL to the clipboard.

```
  import pyperclip
def shorten_url():
     s = pyshorteners.Shortener()
     long_url = url_entry.get()
     try:
         short_url = s.tinyurl.short(long_url)
         output_label.config(text=short_url)
         pyperclip.copy(short_url)
     except Exception as e:
         output_label.config(text="Error shortening URL")
```

Step 6: Running the Application

- Main Loop: Ensured the Tkinter main loop keeps running to display the GUI

```
  root.mainloop()
```

Step 7: Refinements and Debugging

- User Interface: Improved the layout and styling of the GUI for better user experience.

- **Validation**: Added validation to check for empty input and display appropriate messages.

- Error Handling:Implemented error handling to manage invalid URLs and network issues gracefully.

-Testing: Conducted thorough testing to ensure the application works as expected and handles edge cases.

Week 3 Project Report

This Week 3 project report is a detailed record of the project completed during this time. It includes important information such as the project's title, the technology used, the time spent, challenges faced, and a step-by-step explanation of how the project was developed.

Project Details

1. Project Name:

Web Scraping

2. Technology Used:

- **Programming Language**: Python
- **Libraries/Frameworks:** Tkinter (for GUI development), requests (for making HTTP requests), BeautifulSoup (for parsing HTML)

3. **Time Invested and Challenges:**

- Time Invested: Approximately 22 hours over the week.
- Challenges:
  - Understanding the structure of different websites and dealing with inconsistent HTML structures.
  - Handling HTTP request errors and managing rate limits to avoid being blocked by websites.
  - Extracting, cleaning, and organizing data in a meaningful way.

4. Development Process:

Step 1: Project Conception

- Objective: To create a graphical user interface (GUI) application that allows users to scrape data from websites and save the readable content to a text file.

- Initial Research: Reviewed documentation and tutorials on Tkinter, requests, and BeautifulSoup to understand their functionalities.

Step 2: Setting Up the Development Environment

- Tools Installed: Installed Python, Tkinter (included in standard Python distribution), requests, and BeautifulSoup using pip.

```
  pip install requests beautifulsoup4
```

Step 3: Basic GUI Layout

- Window Creation: Created a basic window using Tkinter.

```
  import tkinter as tk
  from tkinter import messagebox

  root = tk.Tk()
  root.title('Web Scraping GUI')
  root.geometry('400x200')
```

- Widgets: Added entry fields for input URL and buttons for actions.

```python
  url_label = tk.Label(root, text='Enter URL:')
  url_label.pack(pady=10)

  url_entry = tk.Entry(root, width=50)
  url_entry.pack(padx=20, pady=5)

  scrape_button = tk.Button(root, text='Scrape and Save',
command=scrape_and_save)
  scrape_button.pack(pady=10)
```

Step 4: Web Scraping Logic

- Scraping Function: Implemented a function to scrape the URL content using `requests` and `BeautifulSoup`.

```
  import requests
  from bs4 import BeautifulSoup

  def scrape_and_save():
    url = url_entry.get()
    if not url:
      messagebox.showwarning('Warning', 'Please enter a valid URL')
      return

    try:
      r = requests.get(url)
```

```
        r.raise_for_status()
        soup = BeautifulSoup(r.content, 'html.parser')

        readable_content = []

        for tag in soup.find_all(['p', 'h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'ul', 'ol', 'li',
'table']):
            if tag.name == 'table':
                table_rows = tag.find_all('tr')
                table_content = []
                for tr in table_rows:
                    row_content = []
                    for td in tr.find_all(['td', 'th']):
                        row_content.append(td.get_text(strip=True))
                    table_content.append('\t'.join(row_content))
                readable_content.append('\n'.join(table_content))
            elif tag.name in ['ul', 'ol']:
                list_items = tag.find_all('li')
                list_content = [li.get_text(strip=True) for li in list_items]
                readable_content.append('\n'.join(list_content))
            else:
                readable_content.append(tag.get_text(strip=True))

        output_file = 'output.txt'
        with open(output_file, 'w', encoding='utf-8') as file:
            file.write('\n\n'.join(readable_content))
        messagebox.showinfo('Success', f'Readable content saved to
{output_file}')


    except requests.exceptions.RequestException as e:
        messagebox.showerror('Error', f'Error fetching URL:\n{str(e)}')
```

Step 5: Running the Application

- Main Loop: Ensured the Tkinter main loop keeps running to display the GUI.

```
 root.mainloop()
```

Step 6: Refinements and Debugging

- **User Interface**: Improved the layout and styling of the GUI for better user experience.

- **Validation**: Added validation to check for empty input and display appropriate messages.

- **Error Handling**: Implemented error handling to manage invalid URLs and network issues gracefully.

- **Testing**: Conducted thorough testing on multiple websites to ensure the application works reliably under different scenarios.

Assignment 1:

1. Assignment Title:

Basic Understanding of Python

2. Tasks Completed:

- Development Roadmap Design: Created a comprehensive roadmap outlining the steps for learning and mastering Python. The roadmap included key milestones, recommended resources, and a timeline for achieving proficiency in Python programming.

- Python Web Development Framework Exploration: Researched and explored various Python web development frameworks such as Flask, Django, and Pyramid. Created a detailed comparison in both PPT and Word format, highlighting the features, advantages, and use cases of each framework.

- Create a Simple Game: Developed a simple number guessing game using Python. The game involves the computer randomly selecting a number within a specified range, and the player has to guess the number. The game provides feedback on whether the guessed number is too high or too low, and continues until the player correctly guesses the number. This project helped in understanding the fundamentals of Python programming, including control structures, functions, and basic input/output operations.

3. Presentation Topics (if applicable):

- Python Programming Overview: Provided an introduction to Python, its history, and its significance in the programming world.

- Learning Path and Resources: Discussed various learning resources, including books, online courses, and practice platforms, to help beginners start their Python journey.

- Web Development Frameworks: Covered the key features and use cases of Flask, Django, and Pyramid. Highlighted the pros and cons of each framework and provided recommendations based on different project requirements.

- Simple Game Development: Presented the process of creating a simple number guessing game, including planning, coding, testing, and debugging. Provided a live demo of the game and discussed the learning outcomes from this project.

Assignment 2:

1. Assignment Title:

Python Libraries Overview

2. Tasks Completed:

- Python Libraries Overview: Created a comprehensive overview of essential Python libraries in both Word and PPT formats. The overview covered libraries for various domains such as data analysis (pandas, NumPy), web development (Flask, Django), machine learning (scikit-learn, TensorFlow), and others. Each library was described with its key features, common use cases, and examples of code snippets.

- Exploration of the Python Ecosystem: Conducted an in-depth exploration of the Python ecosystem, highlighting its diverse range of applications and the extensive community support. This included a detailed analysis of the Python Package Index (PyPI), popular Python libraries, frameworks, tools, and resources for different fields such as web development, data science, automation, and more. The findings were documented in both Word and PPT formats.

3. Presentation Topics (if applicable):

- Python Libraries: Presented key features and use cases of essential Python libraries for various domains, supported by code snippets and examples.

- Python Ecosystem: Discussed the breadth and diversity of the Python ecosystem, emphasizing the extensive range of libraries, frameworks, and tools available for different applications.

Assignment Title:

Calculator App

Tasks Completed:

- Developed a Calculator App: Created a simple calculator application using Python. The app supports basic arithmetic operations such as addition, subtraction, multiplication, and division. The development process included designing the user interface, implementing the core functionality, and testing the app to ensure it works correctly.

- Video Submission: Recorded a video demonstration of the calculator app, showcasing its features and functionality. The video was uploaded to Google Drive for submission.

Presentation Topics (if applicable):

- Overview of the Calculator App: Provided a detailed walkthrough of the calculator app, including its design, functionality, and implementation.

- Demonstration: Showed the calculator app in action, highlighting its features and usability. Discussed the development process and key learning outcomes from the project.

Assignment 3:

1. Assignment Title:

Python Data Structures

2. Tasks Completed:

- Data Structures Overview: Created a comprehensive overview of Python data structures, including lists, tuples, sets, dictionaries, and more complex structures like stacks, queues, linked lists, trees, and graphs. The overview was documented in both Word and PPT formats, providing detailed descriptions, syntax, and examples of each data structure.

- Data Structure Use Cases: Explored various use cases for each Python data structure, illustrating how and when to use them effectively. The use cases included examples from different domains such as web development, data analysis, and algorithms. This information was compiled in both Word and PPT formats.

3. Presentation Topics (if applicable):

- Python Data Structures: Presented the key features, advantages, and limitations of different Python data structures. Provided examples and scenarios where each data structure is most applicable.

- **Practical Use Cases**: Discussed real-world examples and use cases for various data structures, demonstrating their practical applications in solving specific problems.

Assignment Title:

Python Software Development Lifecycle

Tasks Completed:

- Python Software Development Lifecycle: Created a detailed document (DOCX) and presentation (PPT) outlining the software development lifecycle (SDLC) specifically tailored for Python projects. This included phases such as requirements analysis, design, implementation, testing, deployment, and maintenance. Emphasized best practices and methodologies like Agile and DevOps, and discussed tools and techniques relevant to Python development.

Presentation Topics (if applicable):

- SDLC Phases: Provided an overview of each phase in the software development lifecycle, highlighting the key activities and deliverables.

- **Best Practices in Python Development:** Discussed best practices for Python development, including code quality, version control, continuous integration, and testing.

- **Tools and Techniques:** Presented various tools and techniques used in Python development to enhance productivity and ensure project success.

Assignment 4:

1. Assignment Title:

Python Database Interaction

2. Tasks Completed:

- Python Database Interaction: Created a detailed document (DOCX) and presentation (PPT) covering the fundamentals of interacting with databases using Python. This included an introduction to SQL and NoSQL databases, an overview of popular database management systems (DBMS) like MySQL, PostgreSQL, SQLite, and MongoDB, and step-by-step guides on connecting to these databases using Python libraries such as sqlite3, SQLAlchemy, and PyMongo. Additionally, the document and presentation covered CRUD (Create, Read, Update, Delete) operations, transaction management, and best practices for database interaction in Python projects.

3**. Presentation Topics (if applicable):**

- Database Fundamentals: Provided an overview of SQL and NoSQL databases, explaining their differences and use cases.

- Python DBMS Libraries: Discussed popular Python libraries for database interaction and demonstrated how to use them to connect to and manipulate databases.

- Practical Examples: Included practical examples of CRUD operations and transaction management using Python.

Assignment Title:

Python AI and Chatbots

Tasks Completed:

- Python AI and Chatbots: Developed a comprehensive document (DOCX) and presentation (PPT) that delved into the basics of artificial intelligence (AI) and chatbots, with a focus on their implementation using Python. The document and presentation covered topics such as natural language processing (NLP), machine learning (ML) algorithms, and frameworks like TensorFlow and Keras. Detailed steps were provided on building a simple chatbot using libraries like ChatterBot and NLTK, including training the chatbot with a dataset, implementing conversation logic, and integrating the chatbot into a user interface.

3. Presentation Topics (if applicable):

- Introduction to AI and Chatbots: Explained the concepts of artificial intelligence and chatbots, and their relevance in today's technology landscape.

- NLP and ML in Python: Discussed the basics of natural language processing and machine learning, and how they are used in developing intelligent chatbots.

- **Building a Chatbot:** Presented a step-by-step guide to creating a simple chatbot using Python libraries, including data preparation, training, and integration. Provided a live demo of the chatbot and discussed potential improvements and future enhancements.