Summer Internship Report

Mastering C++ : From Object Oriented Programming to Dynamic Programming

(From June 10, 2025 to July 16, 2025)

SUBMITTED BY:

Name of student: Akash yadav

Registration Number: 12310801

To Whom It May Concern,

I, Akash yadav, bearing the Registration Number 12310801, a diligent student pursuing a Bachelor of Technology degree in Computer Science Engineering, hereby solemnly declare that the work detailed herein represents an original undertaking. This project, titled "Mastering C++: From object -oriented Programming to Dynamic Programming" was meticulously completed by me during the months of June 2025 and July 2025.

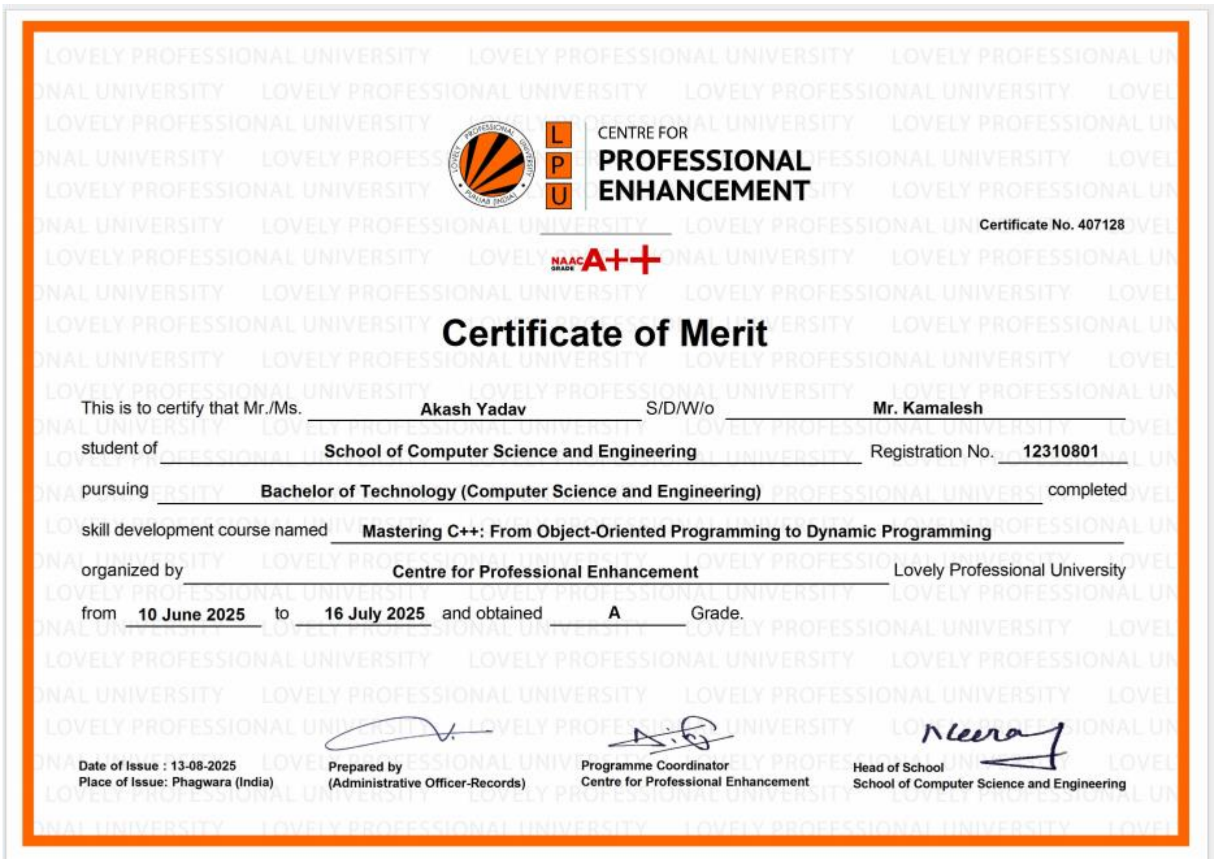
This comprehensive endeavor was undertaken as a crucial component for the partial fulfillment of the academic requirements set forth for the award of my degree. The work reflects my dedicated efforts in understanding, analyzing, and implementing various problem-solving methodologies within the critical domains of OOPs and Dynamic Programming, which are foundational to the field of Computer Science.

I attest that all aspects of this project, from conceptualization to execution and documentation, are a direct result of my own independent research, analysis, and programming efforts. The insights gained and solutions developed during this period contribute to my overall academic and practical understanding of complex computational challenges.

Akash yadav
(Registration Number: 12310801)

Dated: August 23, 2025

Training Certificate



CENTRE FOR PROFESSIONAL ENHANCEMENT

Certificate No. 407128

NAAC GRADE A++

Certificate of Merit

This is to certify that Mr./Ms. **Akash Yadav** S/D/W/o **Mr. Kamalesh**
student of **School of Computer Science and Engineering** Registration No. **12310801**
pursuing **Bachelor of Technology (Computer Science and Engineering)** completed
skill development course named **Mastering C++: From Object-Oriented Programming to Dynamic Programming**
organized by **Centre for Professional Enhancement** Lovely Professional University
from **10 June 2025** to **16 July 2025** and obtained **A** Grade.

Date of Issue : 13-08-2025
Place of Issue: Phagwara (India)

Prepared by
(Administrative Officer-Records)

Programme Coordinator
Centre for Professional Enhancement

Head of School
School of Computer Science and Engineering

1. INTRODUCTION

The **Versioned Document Tracker** is a C++ application developed to provide an efficient mechanism for tracking changes made to a document. It incorporates essential features such as **Undo** and **Redo**, which allow users to navigate across different versions of their work seamlessly. By simulating the functionality of modern text editors, this project demonstrates how document state management can be handled programmatically.

The implementation of this system relies heavily on **fundamental data structures**, including stacks and linked lists, which play a vital role in storing and managing document versions. Additionally, the project makes use of **dynamic memory management** to optimize storage and retrieval of data, ensuring smooth performance during multiple operations. These components highlight the practical applications of concepts learned in data structures and algorithms.

Beyond the technical implementation, the project also emphasizes the importance of **Object-Oriented Programming (OOP)** principles in real-world problem solving. By designing modular classes and applying inheritance, encapsulation, and abstraction, the project not only delivers a functional solution but also strengthens the developer's skills in software design. This combination of data structures, algorithms, and OOP concepts positions the project as a strong learning exercise and a practical application of C++ programming.

2. OBJECTIVE OF THE PROJECT

- To **design and implement** a lightweight and efficient version control system for managing document changes.
- To provide robust **Undo and Redo functionalities** by utilizing appropriate data structures such as stacks and linked lists.
- To **simulate a simplified document tracking mechanism** that resembles the versioning features of modern text editors.
- To enhance practical understanding and strengthen mastery of **C++ concepts** including classes, inheritance, dynamic memory management, and algorithmic optimization.

3. SCOPE OF THE PROJECT

- Provides comprehensive **version history management** for documents, enabling users to track and navigate through all changes made over time.
- Supports multiple **Undo and Redo operations** in a seamless and efficient manner, allowing users to revert or reapply changes without data loss.
- Offers a solid foundation to be **extended into a fully functional mini text editor**, supporting more advanced editing capabilities.
- Can be **integrated into larger collaborative tools or platforms**, enhancing team productivity by maintaining individual version histories.
- Allows for future enhancements such as **file storage integration**, enabling persistent saving of document states to disk.
- Can incorporate additional features like **timestamps**, **version labels**, or **user annotations** to improve traceability and context for each version.
- Has the potential to support **multi-user environments**, where multiple contributors can edit and track document versions concurrently.

---

4. REQUIREMENT SPECIFICATIONS

Hardware Requirements

- Processor: Intel i3 or above
- RAM: Minimum 2 GB
- Storage: 100 MB

Software Requirements

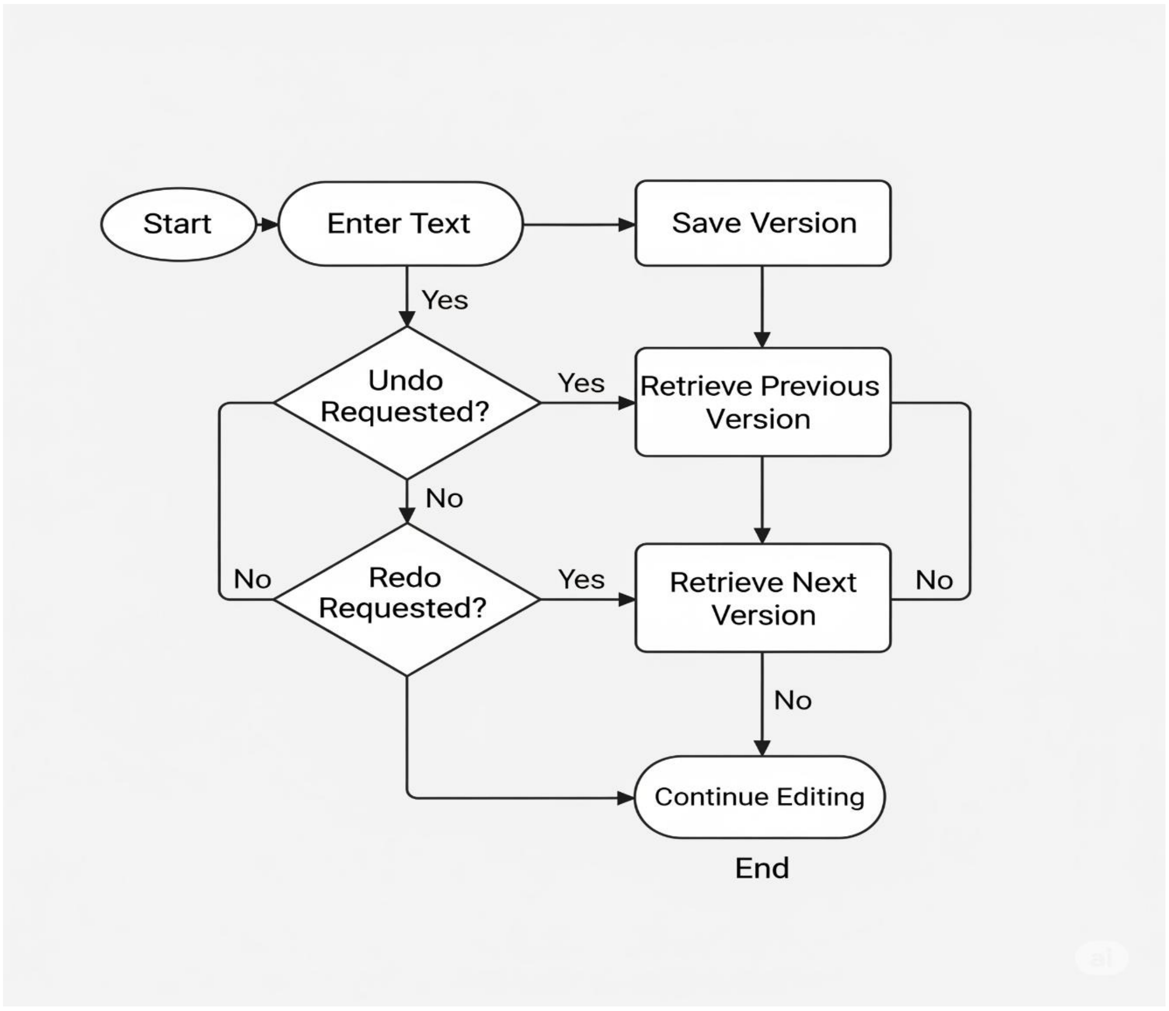
- Programming Language: C++ (C++11 or above)
- IDE: Code::Blocks / Dev-C++ / Visual Studio Code
- Operating System: Windows / Linux

5. MODULES OF THE APPLICATION

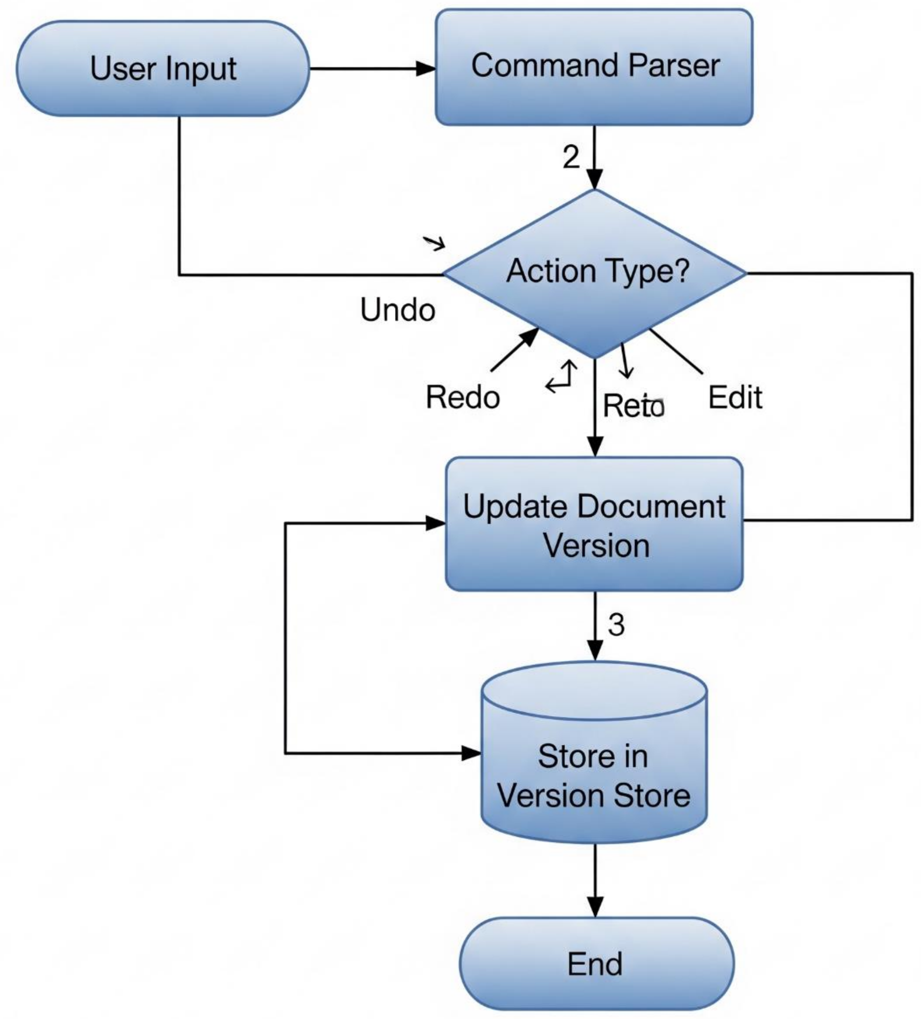
1. **Document Module** – Stores and manages the current state of the document.
2. **Version Control Module** – Maintains undo and redo stacks to track versions.
3. **User Interface Module (Console-based)** – Accepts and processes user commands.
4. **File Handling Module (Optional)** – Provides persistent storage of versions on disk.

---

6. Flow Chart



7. Data Flow Diagram



8. ARCHITECTURE

Frontend Layer (Terminal-Based Interface):

The user interacts with the system directly through the terminal. This simple and lightweight interface allows users to type commands such as adding text, performing undo, performing redo, and viewing the current state of the document. Since the project runs entirely in the terminal, it emphasizes functionality over graphical design while still ensuring a user-friendly workflow.

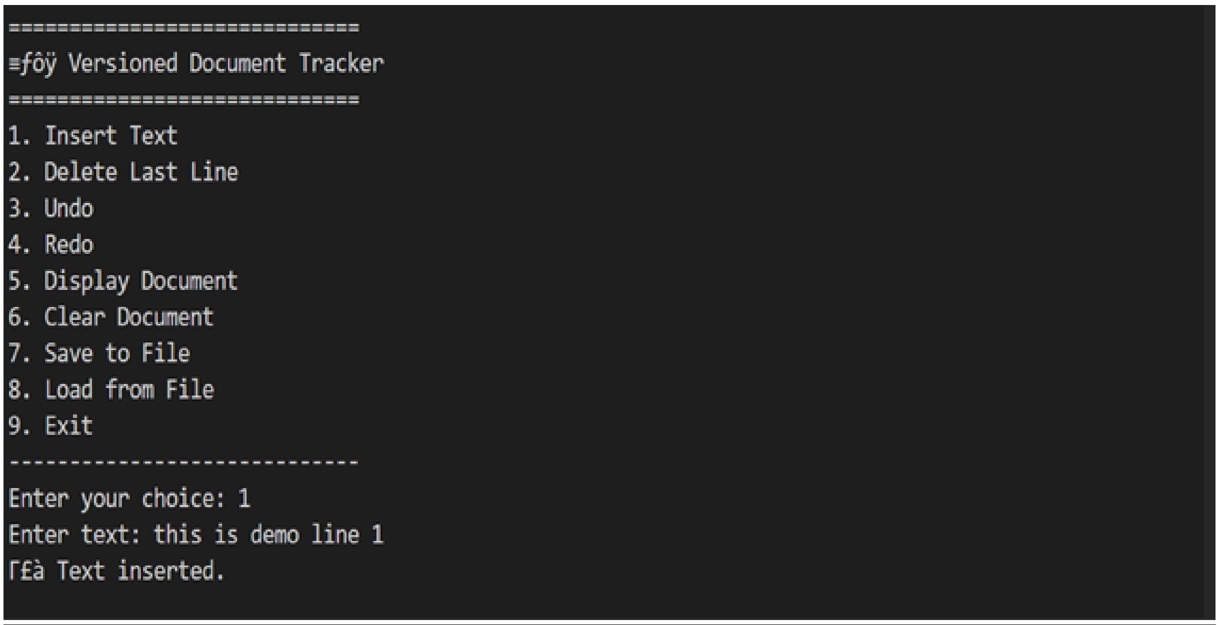
Application Logic Layer (C++ Classes):

This layer contains the **core logic of the application**, developed using Object-Oriented Programming principles. Dedicated classes manage tasks such as handling document edits, processing user commands, and controlling the undo/redo operations. The modular design ensures that each class is responsible for a specific function, making the project easy to maintain and extend.

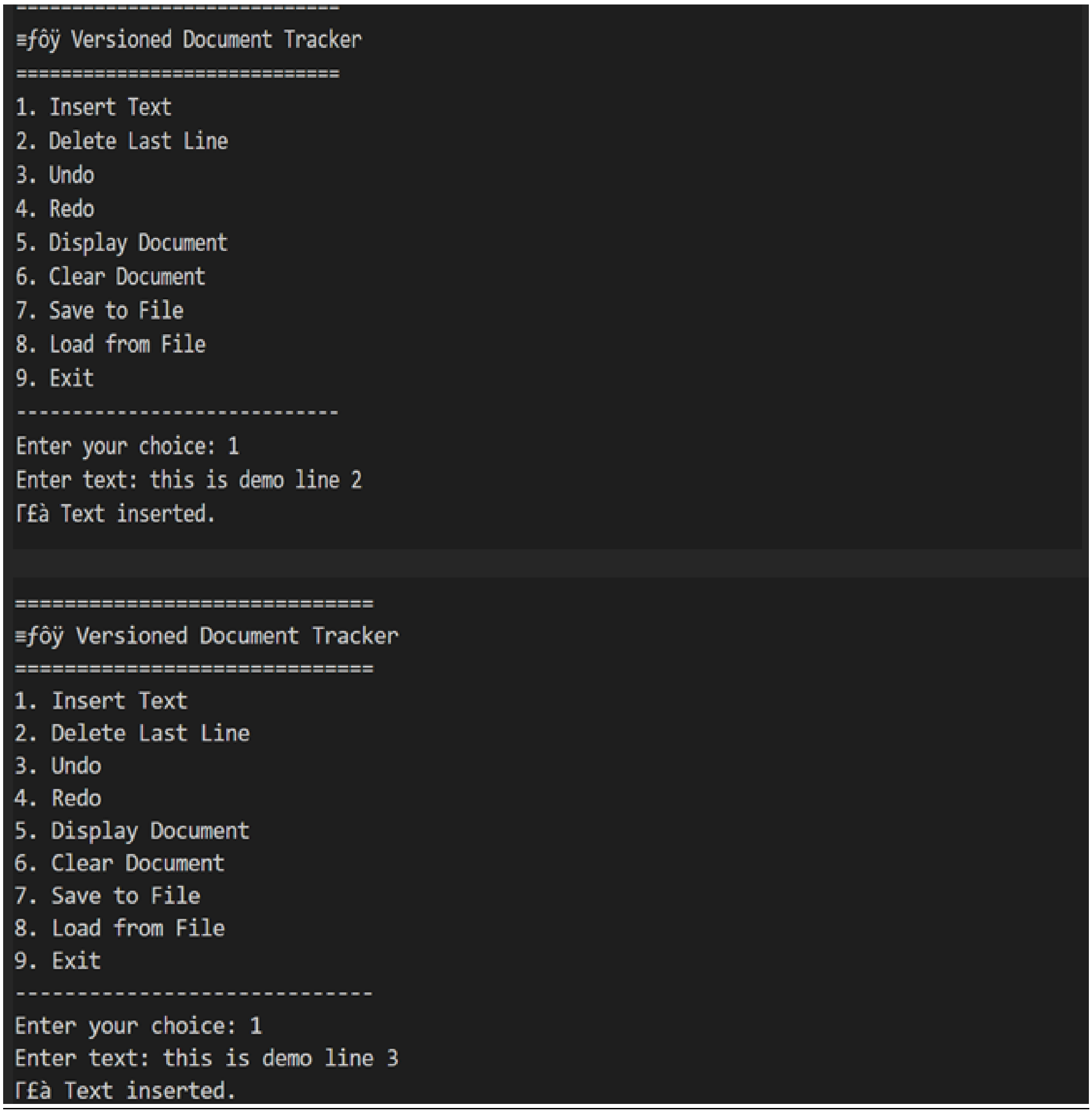
Data **Storage Layer (Stacks and Linked Lists):**

- **Undo Stack:** Stores previous document states, allowing the system to revert changes.
- **Redo Stack:** Stores undone states, enabling the reapplication of actions.
  Dynamic memory allocation and linked lists are used to handle variable-sized inputs, ensuring efficient memory utilization while managing multiple versions of the document.

9. SCREENSHOTS OF PROJECT



```
==============================
≡ƒôÿ Versioned Document Tracker
==============================
1. Insert Text
2. Delete Last Line
3. Undo
4. Redo
5. Display Document
6. Clear Document
7. Save to File
8. Load from File
9. Exit
------------------------------
Enter your choice: 1
Enter text: this is demo line 1
Γ£à Text inserted.
```



```
==============================
≡ƒôÿ Versioned Document Tracker
==============================
1. Insert Text
2. Delete Last Line
3. Undo
4. Redo
5. Display Document
6. Clear Document
7. Save to File
8. Load from File
9. Exit
------------------------------
Enter your choice: 1
Enter text: this is demo line 2
Γ£à Text inserted.
```

```
==============================
≡ƒôÿ Versioned Document Tracker
==============================
1. Insert Text
2. Delete Last Line
3. Undo
4. Redo
5. Display Document
6. Clear Document
7. Save to File
8. Load from File
9. Exit
------------------------------
Enter your choice: 1
Enter text: this is demo line 3
Γ£à Text inserted.
```

10. CONCLUSION

The project successfully demonstrates the implementation of a **Versioned Document Tracker with Undo/Redo operations**. It highlights the importance of **efficient data structures** in real-world applications such as text editors.

This project provided hands-on experience in **Object-Oriented Programming, pointers, stacks, linked lists, and dynamic memory management in C++**. Beyond strengthening theoretical knowledge, it also offered practical insight into how algorithms can enhance user experience in version control systems.

9. REFERENCES

1. https://cplusplus.com/reference/stack/stack/

2. C++ STL Documentation

3. Geeks For Geeks website

4. LPU Course Notes

5. W3 school website