INDUSTRIAL TRAINING

REPORT/SUMMER TRAINING REPORT

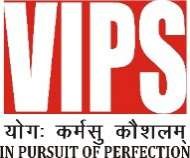
# MASTER DATA STRUCTURES & ALGORITHMS USING C++

Submitted in partial fulfilment of the

Requirements for the award of Degree of

Bachelor of Technology in

CSE



VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS

Grade A++ Accredited Institution by NAAC

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

NNN

Submitted By

Name: Ishaan Jain

University Roll No.: 0611770272

Semester: VII

Academic Year: 2025-26

i

# **DECLARATION**

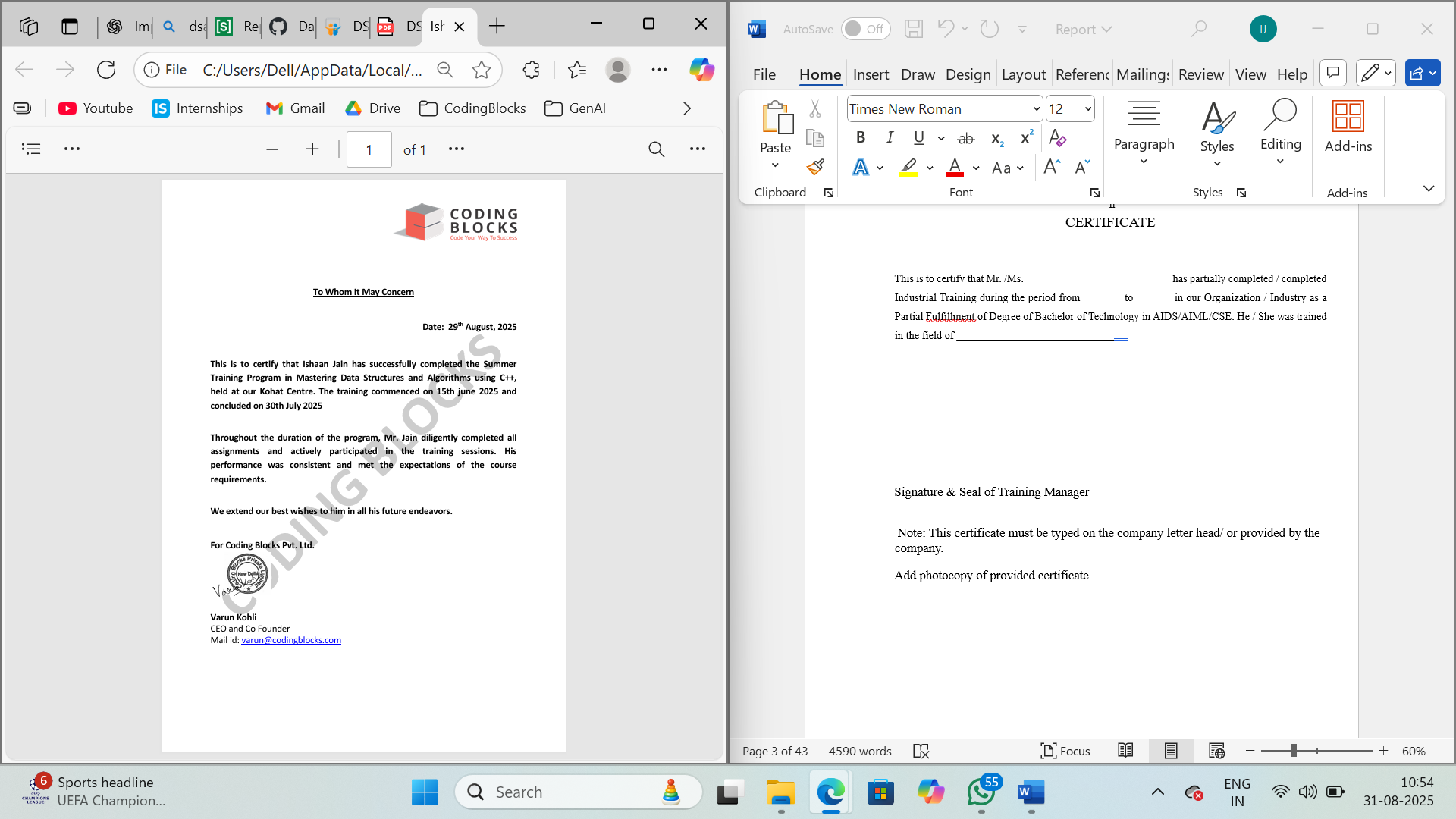
# I hereby declare that the Industrial / Summer Training Report entitled MASTER DATA STRUCTURES & ALGORITHMS USING C++ is an authentic record of my own work as requirements of Industrial Training during the period from 15st June to 30th July for the award of degree of B.Tech. CSE School of Engineering and Technology, VIPS-TC.

Date: 03-09-2025 Ishaan Jain

06117702722

ii

# **CERTIFICATE**



# iii

# **ACKNOWLEDGEMENT**

It is with great pleasure and sincere gratitude that I present this summer training report, the result of my learning experience at **Coding Blocks**, undertaken as a part of my academic and professional development journey. This training has been an enriching phase in my pursuit of mastering **Data Structures and Algorithms using C++**, and I am deeply indebted to all those who guided and supported me throughout this endeavour. This journey would not have been possible without the collective support and guidance of several individuals and institutions, to whom I am profoundly thankful.

First and foremost, I would like to express my heartfelt gratitude to **Mr. Mayank Jha Sir** for his constant mentorship and guidance during my training. His clear teaching methodology, deep subject expertise, and patient encouragement have been invaluable in strengthening my problem-solving skills and sharpening my conceptual understanding. The insights I gained under his supervision will continue to guide me in both my academic and professional pursuits.

I also extend my sincere thanks to the entire team at **Coding Blocks** for providing such a well-structured and interactive learning environment. The comprehensive curriculum, practice-driven approach, and continuous feedback significantly enhanced my ability to apply theoretical concepts to practical problems.

I am also deeply indebted to my academic institution, Vivekananda Institute of Professional Studies - Technical Campus (VIPS-TC). I extend my sincere thanks to the faculty and administration for their unwavering support. The foundational knowledge and principles instilled in me during my coursework provided the essential groundwork for tackling the challenges of this project. Their encouragement and academic guidance have always been a source of motivation for me to strive for excellence.

Finally, I am deeply grateful to my family for their unwavering support and patience, which allowed me to focus completely on this invaluable learning experience.

iv

# **ABOUT THE COMPANY**

**Coding Blocks** is a premier coding education company based in India, dedicated to bridging the gap between academic learning and industry requirements by providing high-quality training in programming and software development. Since its inception, Coding Blocks has established itself as one of the leading institutions for aspiring programmers, offering structured programs designed to transform beginners into industry-ready professionals.

The company specializes in delivering comprehensive courses across various domains, including:

* **Data Structures and Algorithms (DSA):** Strengthening problem-solving abilities through rigorous practice in languages such as C++ and Java.
* **Web Development:** Covering both frontend and backend technologies to build scalable, modern web applications.
* **Mobile App Development:** Focusing on Android and iOS development using popular frameworks and tools.
* **Machine Learning and Artificial Intelligence:** Equipping students with the knowledge and skills to work with cutting-edge AI and data-driven technologies.
* **Interview Preparation and Competitive Programming:** Providing targeted guidance to excel in coding competitions and technical job interviews.

Coding Blocks is well known for its highly qualified instructors, many of whom are industry experts and alumni from prestigious institutions such as IITs and IIITs. The company emphasizes hands-on learning through coding assignments, projects, hackathons, and doubt-solving sessions, ensuring that students gain both theoretical clarity and practical expertise.

With its commitment to nurturing the next generation of software engineers, **Coding Blocks has empowered thousands of students and professionals to succeed in the tech industry**, building strong careers at top multinational corporations, startups, and global tech companies.

v

ABOUT THE ROLE: TRAINEE

During my summer training at **Coding Blocks**, I had the privilege of undertaking an intensive program focused on mastering **Data Structures and Algorithms (DSA) using C++**. This role was specifically designed to strengthen my problem-solving skills, algorithmic thinking, and coding proficiency, bridging the gap between classroom theory and practical application in the field of computer science.

The training began with a rigorous foundation-building phase, where I revised core C++ concepts and explored advanced programming techniques. Under the guidance of **Mr. Mayank Jha Sir**, I worked on a structured curriculum that included:

* **Arrays, Strings, and Recursion:** Strengthening fundamental problem-solving approaches.
* **Linked Lists, Stacks, and Queues:** Understanding dynamic memory management and linear data structures.
* **Trees and Graphs:** Gaining insights into hierarchical and network-based data structures with applications in real-world problems.
* **Dynamic Programming and Greedy Algorithms:** Developing optimization strategies for solving complex computational problems efficiently.
* **Sorting, Searching, and Hashing Techniques:** Mastering efficient data handling and retrieval mechanisms

Throughout the internship, I solved numerous practice problems, participated in coding challenges, and implemented algorithms from scratch to gain a deeper understanding of their working principles. This hands-on approach not only improved my **analytical and coding skills** but also provided me with a competitive edge in technical interviews and contests.

By the end of the program, I had acquired the ability to break down complex problems into smaller, manageable parts and design efficient solutions, which will serve as a strong foundation for my future endeavors in software development and competitive programming.

# v

# **TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| Chapter | Title | Page No. |
| - | Certificate by Company | i |
| - | Declaration by Student | ii |
| - | Acknowledgement | iii |
| - | About the Company | iv |
| - | Table of Contents | V |
| - | List of Figures | vi |
| - | Abstract | vii |
| 1 | Introduction to Project | 11 |
|  | 1.1 Background and Purpose of the Project | 11 |
|  | 1.2 Project Objectives | 11 |
|  | 1.3 Scope of the Project | 12 |
| 2 | Tools & Technology Used | 13 |
|  | 2.1 Technology Stack Overview | 13 |
|  | 2.2 Programming Language: C++ | 13 |
|  | 2.3 Integrated Development Environment (IDE) | 13 |
|  | 2.4 Competitive Programming Platforms | 14 |
|  | 2.5 Git & GitHub for Version Control | 14 |
| 3 | Reason for Choosing DSA | 15 |
|  | Use Cases of DSA | 16 |
| 4 | Introduction to Mini Project. | 18 |
|  | 4.1 Synopsis Project | 18 |
|  | 4.2 Objective of the Project | 18 |
|  | 4.3 System Features | 20 |
|  | 4.4 Key Programming Concepts | 21 |
|  | 4.5 System Constraints and Validations | 22 |
|  | 4.6 Educational Value | 22 |
|  | 4.7 Future Enhancements | 22 |
|  | 4.8 Conclusion | 23 |
| 5 | Snapshots and Results | 24 |
|  | 5.1 Linked List Implementation | 24 |
|  | 5.2 Queue Implementation | 27 |
|  | 5.3 Stack Implementation | 28 |
|  | 5.4 Main Functionalities | 29 |
| 6 | Conclusion and Future Scope | 31 |
|  | 6.1 Conclusion of the Project | 31 |
|  | 6.2 Key Learning Outcomes | 31 |
|  | 6.3 Limitations and Future Enhancements | 32 |
|  | 6.4 Practical Project Highlight: Hotel Management System | 33 |
|  | 6.5 Integration with Learning and Future Enhancements | 34 |
| 7 | References | 35 |

vi

**LIST OF FIGURES**

|  |  |
| --- | --- |
| 5.1.1 | Setting up Header Files and State Variables for User Management |
| 5.1.2 | Setting up Linked List for Customer Details and Room Booking |
| 5.1.3 | Function to Book the Room |
| 5.1.4 | Function to Delete the Last or Specific Booking |
| 5.1.5 | Function to Display All Values and Delete All the List |
| 5.2.1 | Setting up Queue for Handling Restaurant |
| 5.2.2 | Function to Add, Delete, and Display Dining Choices |
| 5.2.3 | Queue Class Setup |
| 5.2.4 | Function to Delete Head and Print All Details |
| 5.3.1 | Stack Class with Pop, Push, and Peek Functions |
| 5.4.1 | Choice Selection |
| 5.4.2 | Room Booking Selection (Laundry, Restaurant, Book Room) |
| 5.4.3 | Laundry Selection (Wash, Deliver, Display) |
| 5.4.4 | Restaurant Booking Selection (Reservation, Dining, Display) |
| 5.4.5 | Stack Function Call |

# vii

# **ABSTRACT**

This report highlights the learning and application of **Data Structures and Algorithms (DSA) using C++**, undertaken as the primary focus during my summer training program at **Coding Blocks**. The core objective of this training was to strengthen problem-solving ability, enhance algorithmic thinking, and gain hands-on proficiency in writing efficient, optimized code.

The training followed a structured curriculum, beginning with foundational concepts of C++ programming and gradually progressing towards advanced algorithmic techniques. Key topics covered included **arrays, recursion, linked lists, stacks, queues, trees, graphs, hashing, sorting, searching, greedy algorithms, and dynamic programming**. Each concept was reinforced through extensive coding practice, problem-solving exercises, and algorithm implementation from first principles, ensuring a deep understanding of both theoretical and practical aspects.

In addition to theoretical learning, I engaged in solving numerous real-world coding problems and competitive programming challenges that simulated industry-level problem-solving scenarios. This practice-oriented approach not only improved my coding speed and accuracy but also equipped me with the analytical mindset required to tackle complex computational problems.

The culmination of this training lies in the transformation of my problem-solving skills, the ability to analyze and optimize algorithms, and the confidence to apply these techniques in both academic and professional domains. This experience has laid a strong foundation for my career in software development, preparing me for technical interviews, competitive programming, and future real-world projects.

# **CHAPTER 01: INTRODUCTION**

## **1.1 Background and Purpose of the Project**

In the rapidly evolving field of computer science, **Data Structures and Algorithms (DSA)** form the backbone of efficient problem-solving and software development. From system design and database management to competitive programming and technical interviews, mastery of DSA is considered a fundamental requirement for aspiring software engineers. Proficiency in this domain not only enhances logical and analytical thinking but also equips students to build optimized and scalable solutions to real-world problems.

The **summer training at Coding Blocks** provided me with an invaluable opportunity to strengthen these skills under the structured guidance of industry professionals. The training, led by **Mr. Mayank Jha Sir**, was designed to bridge the gap between academic coursework and practical application by focusing on hands-on coding, problem-solving techniques, and advanced algorithmic concepts using **C++**.

The purpose of this training was twofold: first, to gain a deep understanding of classical and advanced data structures and algorithms; and second, to apply this knowledge through continuous coding practice and problem-solving exercises, thereby preparing for both academic excellence and future professional challenges.

## **1.2 Project Objectives**

The primary objectives of this training were defined to ensure comprehensive learning and practical application. The key goals included:

* To revise and strengthen the fundamentals of **C++ programming**.
* To gain in-depth understanding of **linear and non-linear data structures** including arrays, strings, linked lists, stacks, queues, trees, and graphs.
* To study and implement **algorithmic paradigms**, including divide and conquer, greedy algorithms, and dynamic programming.
* To develop the ability to solve **complex computational problems** with efficiency and accuracy.
* To enhance **competitive programming skills** and prepare for coding interviews by practicing real-world problem statements.
* To build confidence in analyzing problems, designing optimized solutions, and writing clean, maintainable code.

## **1.3 Scope of the Training**

The scope of this training was designed to cover the essential and advanced concepts of data structures and algorithms using C++.

The key areas within scope included:

* **Foundations of C++ programming:** Syntax, functions, recursion, and object-oriented programming basics.
* **Data Structures:** Arrays, strings, linked lists, stacks, queues, trees (binary trees, BSTs, heaps), and graphs.
* **Algorithms:** Searching, sorting, hashing, shortest path, spanning trees, greedy strategies, and dynamic programming.
* **Problem Solving:** Implementation of algorithms and solving practice problems on platforms like HackerRank, Codeforces, and LeetCode, CodeChef.

Functionalities beyond the immediate scope but identified as potential future enhancements include:

* **Advanced topics** such as string algorithms (KMP, Rabin-Karp), segment trees, and Fenwick trees.
* **Specialized competitive programming strategies** like bit manipulation and number theory.
* **Project-based application of DSA**, such as building mini software systems or competitive coding bots.

# CHAPTER 02: TOOLS AND TECHNOLOGIES USED

## **2.1 Technology Stack Overview**

The summer training program at **Coding Blocks** focused on building a strong foundation in **Data Structures and Algorithms (DSA)** using **C++**. The technology stack was selected to provide a balance between powerful programming capabilities, ease of learning, and practical application in both academic and professional settings. The core stack included **C++ as the programming language**, widely recognized for its efficiency and control over system resources, along with modern development tools and competitive programming platforms to simulate real-world problem-solving.

## **2.2 Programming Language: C++**

C++ was the central technology used throughout the training. It is one of the most widely used languages in systems programming, competitive coding, and application development.

Key reasons for choosing Spring Boot:

* **Efficiency and Performance:** Provides direct memory management and faster execution, essential for algorithmic problem-solving.
* **Rich Standard Template Library (STL):** Offers built-in implementations of common data structures (like vectors, maps, sets) and algorithms, helping in rapid development.
* **Versatility:** Supports both procedural and object-oriented paradigms, making it suitable for a wide range of problem types.
* **Industry Relevance:** Highly valued in competitive programming and coding interviews conducted by top tech companies.

## **2.3 Development Environment**

To ensure seamless practice and implementation, modern tools and environments were utilized:

* **Integrated Development Environments (IDE):**
  + *Visual Studio Code* and *CodeBlocks* were primarily used for writing, debugging, and testing C++ programs.
* **Compiler:**
  + *GCC (GNU Compiler Collection)* and *MinGW* were used to compile and run C++ code, ensuring compatibility across systems.
* **Online IDEs and Platforms:**
  + The *Coding Blocks Online IDE* was frequently used for quick practice and code sharing.

## **2.4 Competitive Programming Platforms**

An important component of mastering DSA is solving problems in a time-constrained environment. To practice real-world coding challenges, several online judges were used:

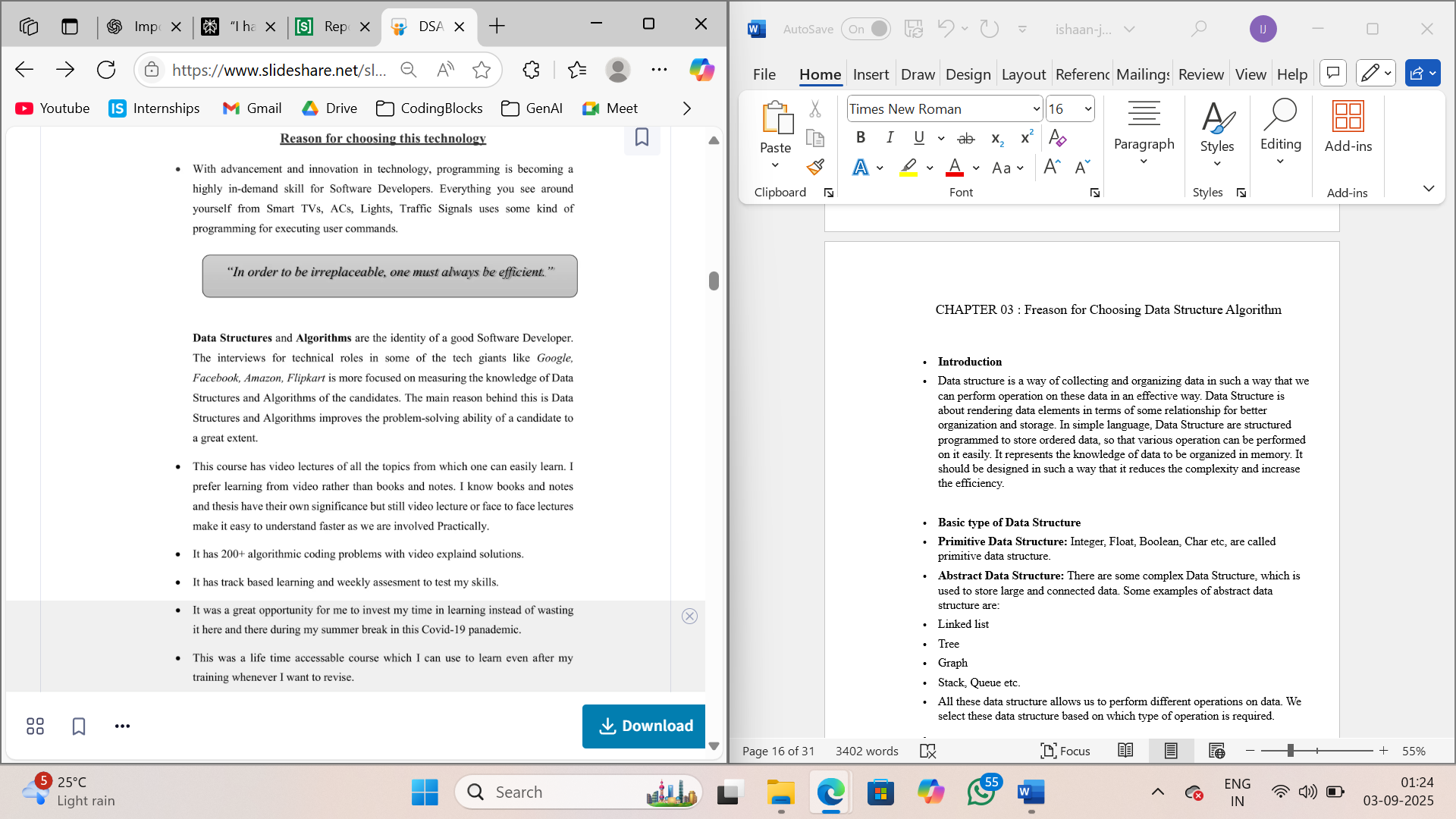
* **HackerRank & HackerEarth** – for practicing beginner to intermediate problems.
* **LeetCode** – for interview preparation and solving company-tagged questions.
* **Codeforces & CodeChef** – for participating in live contests and improving speed.
* **Coding Blocks Portal** – provided curated problems aligned with the training curriculum.

**2.5 Version Control**

To manage projects and maintain code versions:

* **Git & GitHub** were used for version control, allowing easy code tracking, sharing, and collaboration.

# **CHAPTER 03: Reason for Choosing Data Structure Algorithm**

With advancement and innovation in technology, programming is becoming a highly in-demand skill for Software Developers. Everything you see around—from Smart TVs, ACs, Lights, Traffic Signals—uses some kind of programming for executing user commands.

Data Structures and Algorithms are the identity of a good Software Developer. The interviews for technical roles in some of the tech giants like *Google, Facebook, Amazon, Flipkart* are more focused on measuring the knowledge of Data Structures and Algorithms of the candidates. The main reason behind this is Data Structures and Algorithms improve the problem-solving ability of a candidate to a great extent.

* This course has video lectures on all the topics, making learning easy. I prefer learning from video rather than books and notes. While books and thesis have their own significance, video lectures or face-to-face sessions make it easier to understand, especially through practical involvement.
* The course offers 200+ algorithmic coding problems with video-explained solutions.
* It provides track-based learning and weekly assessments to test my skills.
* It was a great opportunity for me to invest time in learning, rather than wasting time during my summer break amid the Covid-19 pandemic.
* This is a lifetime accessible course, allowing me to revise and learn even after my training ends.

**3.1 USE CASES OF DSA**

**1ST USE CASE - PRODUCT BASE COMPANIES INTERVIEWS**

In our daily life, we always go with that person who can complete the task in a short amount of time with efficiency and using fewer resources. The same things happen with these companies. The problem faced by these companies is much harder and at a much larger scale. Software developers also have to make the right decisions when it comes to solving the problems of these companies.

For example, in an interview, you're given a problem to find the sum of first N natural numbers.

One candidate solves it by using a loop like:

Initialize sum = 0

for every natural number n in range 1 to N (inclusive):

  add n to sum

sum is the answer

And you solve it using the sum of first N natural numbers given by the formula:

Sum = N × (N + 1) / 2

Obviously, they will choose you over other one because your solution is more efficient.

Knowledge of data structures like Hash Tables, Trees, Tries, Graphs, and various algorithms goes a long way in solving these problems efficiently and the interviews are more interested in seeing how candidates use these tools to solve a problem. Just like a car mechanic needs the right tool to fix a car and make it run properly, a programmer needs the right tool (algorithm and data structure) to make the software run properly. So, the interviewer wants to find a candidate who can apply the right set of tools to solve the given problem. If you know the characteristics of one data structure in contrast to another you will be able to make the right decision in choosing the right data structure to solve a problem.

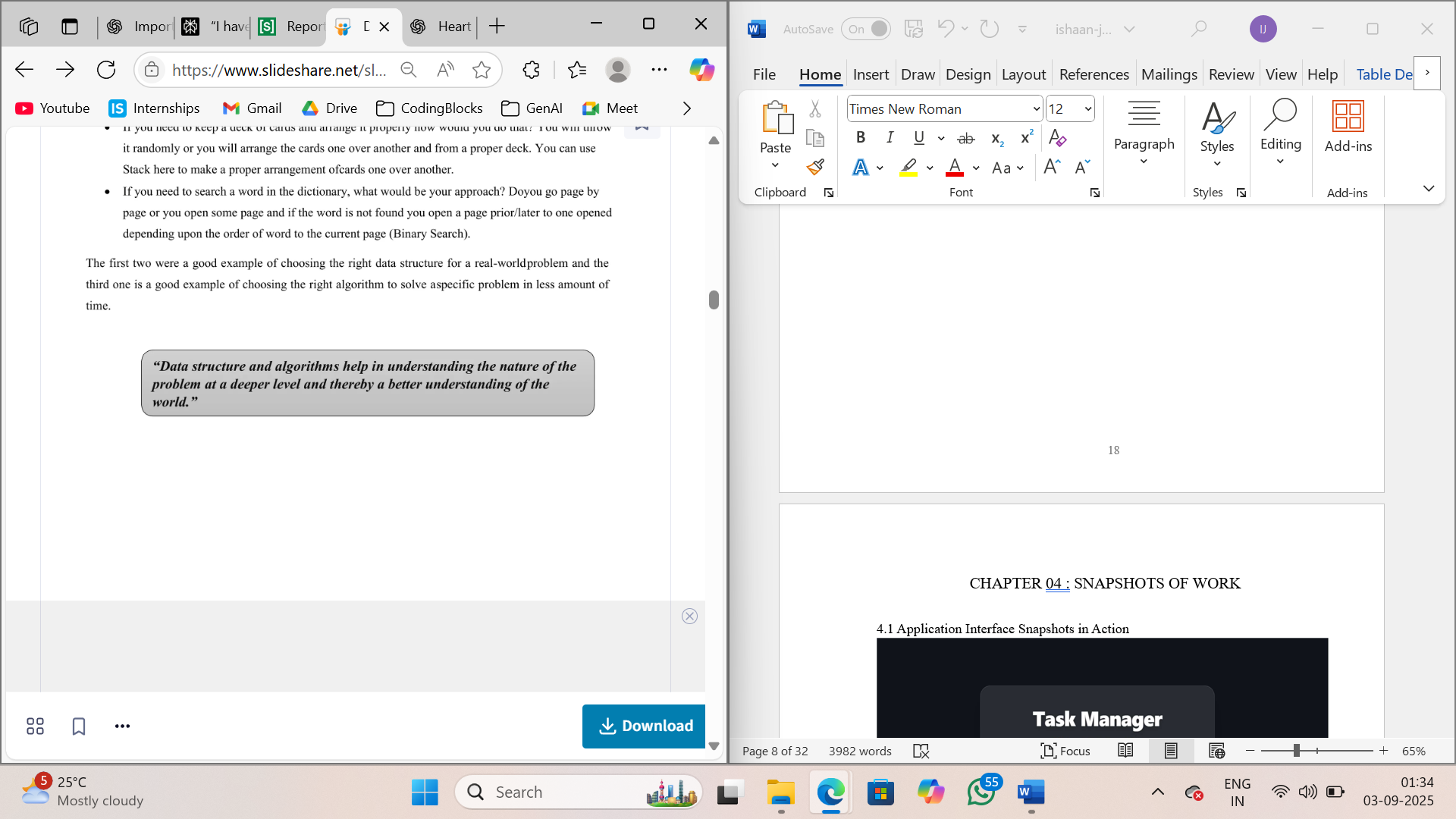
**2nd USE CASE – Solving Real World Complex Problems**

Let's take the example of Library. If you need to find a book on Set Theory from a library, you will go to the math section first, then the Set Theory section. If these books are not organized in this manner and just distributed randomly then it will be frustrating to find a specific book. So, data structures refer to the way we organize information on our computer. Computer scientists process and look for the best way we can organize the data we have, so it can be better processed based on input provided.

A lot of newbie programmers have this question that where we use all the stuff of data structure and algorithm in our daily life and how it’s useful in solving the real-world complex problem. We need to mention that whether you are interested in getting into the top tech giant companies or not, DSA still helps a lot in your day-to-day life.

Let’s consider some examples:

* In Facebook you can represent your friends on Facebook, friends of friends, mutual friends easily by Graph.
* If you need to keep a deck of cards and arrange it properly, how would you do that? You will throw it randomly or you will arrange the cards one over another and form a proper deck. You can use Stack here to make a proper arrangement of cards one over another.
* If you need to search a word in the dictionary, what would be your approach? Do you go page by page or you open some page and if the word is not found you open a page prior/later to one opened depending upon the order of word to the current page (Binary Search).

The first two were a good example of choosing the right data structure for a real-world problem and the third one is a good example of choosing the right algorithm to solve a specific problem in less amount of time.

# **CHAPTER 04: Introduction to Mini Project- The Hotel Management System**

## **4.1 Synopsis Project**

The Hotel Management System is a comprehensive console-based application developed as part of a Data Structures and Algorithms (DSA) project during summer training. This system demonstrates the practical implementation of three fundamental data structures - **Linked Lists**, **Stacks**, and **Queues** - to manage various hotel operations efficiently.

The project simulates real-world hotel management scenarios by providing automated solutions for room booking, laundry services, and restaurant reservations. Each core functionality is built using a specific data structure, showcasing how different data structures can be optimally utilized for different types of operations in a business environment.

**4.2 Objective of The Project**

**Primary Objectives**

1. **Implement Core Data Structures**: Demonstrate practical usage of Linked Lists, Stacks, and Queues in a real-world application
2. **Hotel Operations Management**: Create an integrated system to handle multiple hotel services efficiently
3. **Memory Management**: Implement dynamic memory allocation and deallocation using pointers and classes
4. **User Interface Design**: Develop an intuitive console-based menu system for easy navigation
5. **Data Persistence**: Manage customer information and service requests systematically

**Learning Objectives**

1. **Data Structure Implementation:** Hands-on experience with implementing data structures from scratch
2. **Object-Oriented Programming:** Utilize classes and objects to create modular, reusable code
3. **Algorithm Design:** Develop efficient algorithms for insertion, deletion, and traversal operations
4. **Problem Solving:** Apply appropriate data structures to solve specific business problems.

**4.3 System Features**

**1. Room Booking System (Linked List Implementation)**

* **Add Guest Information:** Register new guests with room number, name, address, and phone
* **Remove Specific Booking:** Cancel booking for a specific room number
* **Remove All Bookings:** Clear all room reservations
* **Display All Bookings:** View complete list of current room reservations
* **Room Validation:** Ensures room numbers are within valid range (1-200)

**Why Linked List?**

* Dynamic memory allocation for unlimited bookings
* Efficient insertion and deletion of guest records
* Sequential access suitable for displaying all bookings

**2. Laundry Service System (Stack Implementation)**

* **Accept Laundry**: Add clothes for washing and pressing (PUSH operation)
* **Deliver Laundry**: Return processed clothes to guests (POP operation)
* **View Pending Orders**: Display all pending laundry requests
* **Quantity Validation**: Limits clothes per order to 50 items maximum
* **Room Validation**: Ensures valid room numbers (1-200)

**Why Stack?**

* LIFO (Last In, First Out) principle matches laundry processing
* Most recent orders are processed first
* Simple push/pop operations for order management

**3. Restaurant Reservation System (Queue Implementation)**

* **Make Reservation**: Book dining slots with room number and meal preference
* **Complete Dining**: Remove guests who have finished dining (DEQUEUE operation)
* **View Reservation Queue**: Display all pending restaurant reservations
* **Menu Display**: Shows available dining options with timings:
* Breakfast: 8am - 11am
* Lunch: 12pm -3pm
* Hi Tea: 4pm – 7pm
* Dinner: 8pm – 2am

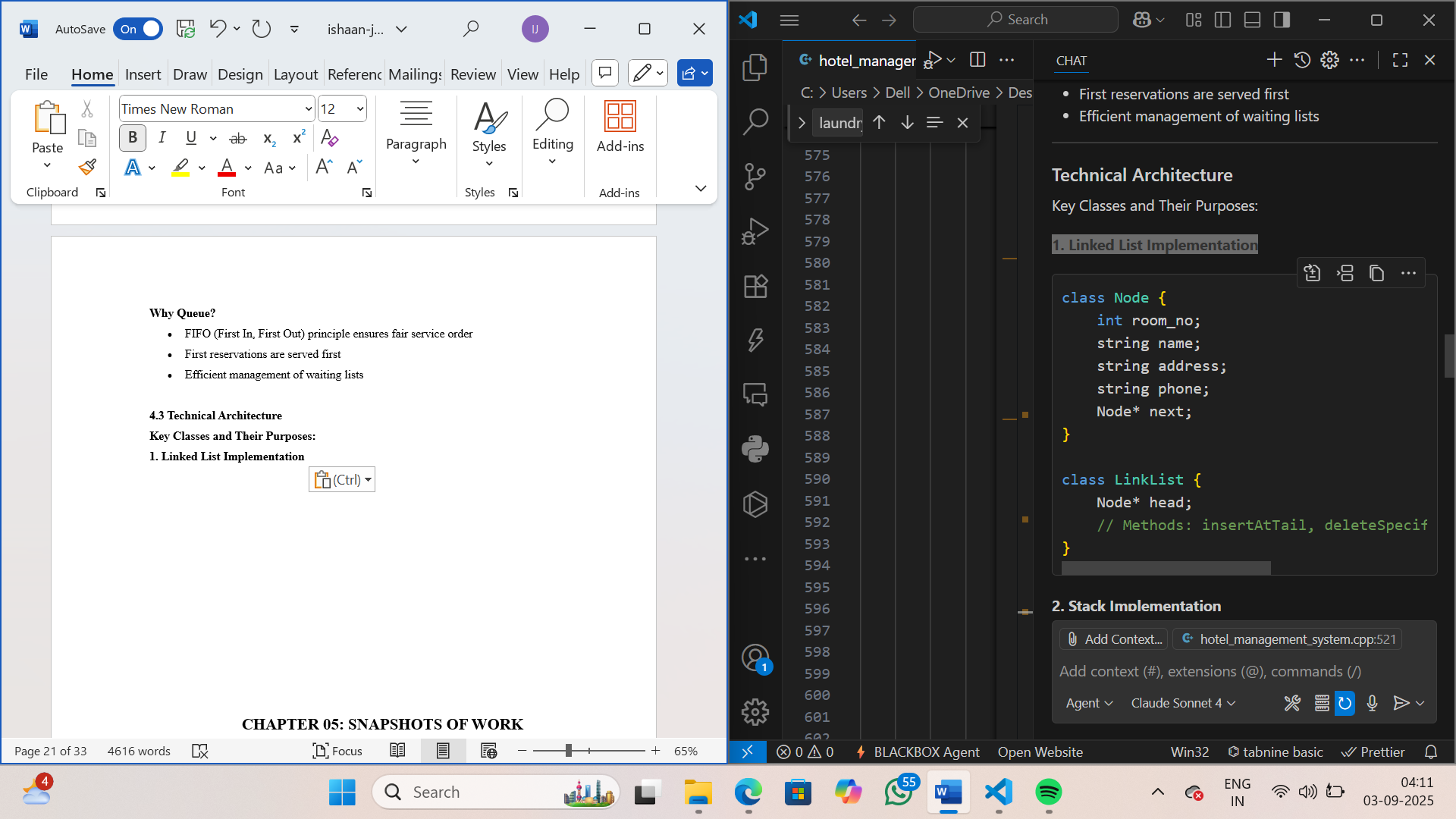
**Why Queue?**

* FIFO (First In, First Out) principle ensures fair service order
* First reservations are served first
* Efficient management of waiting lists

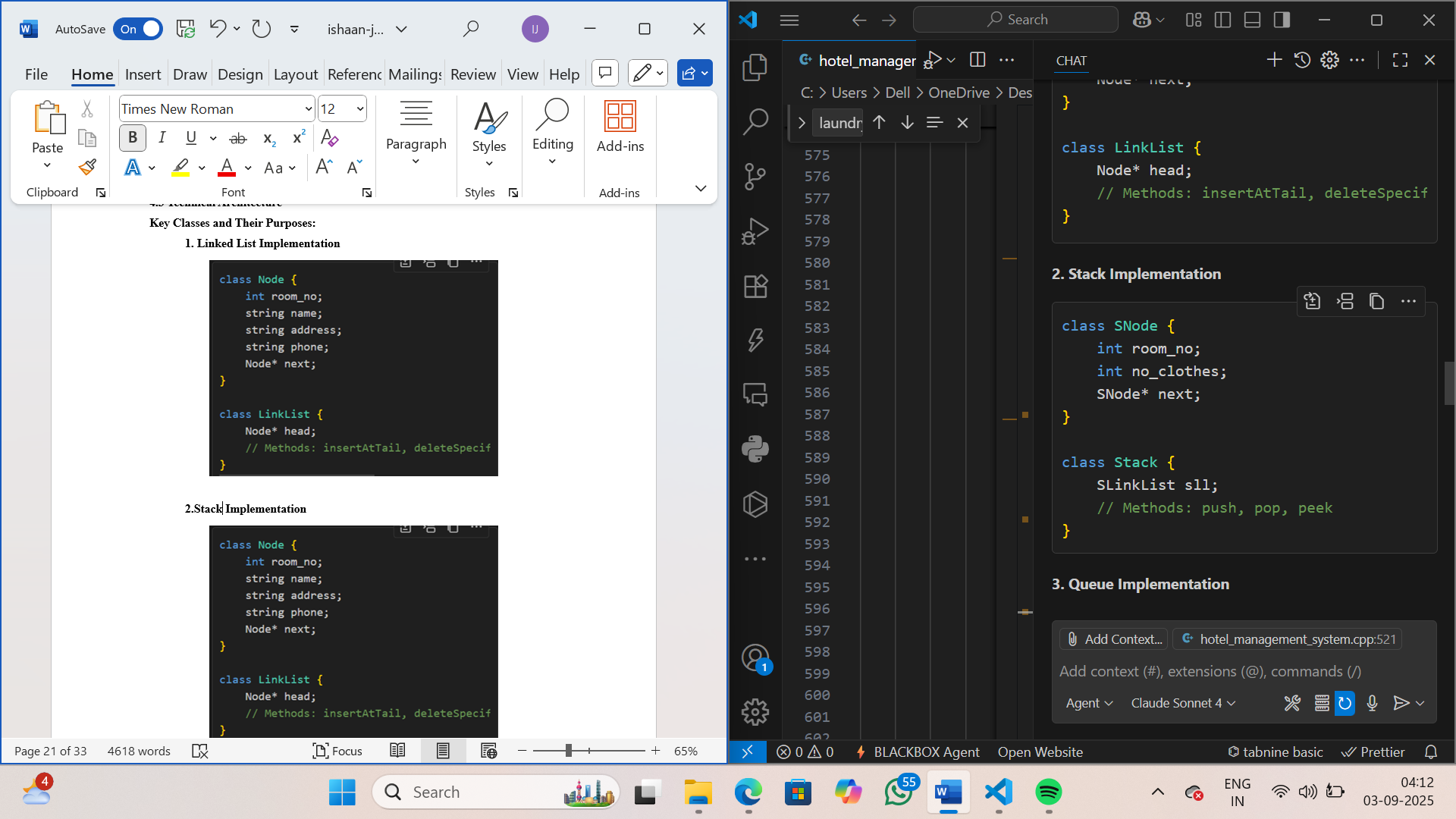
**4.3 Technical Architecture**

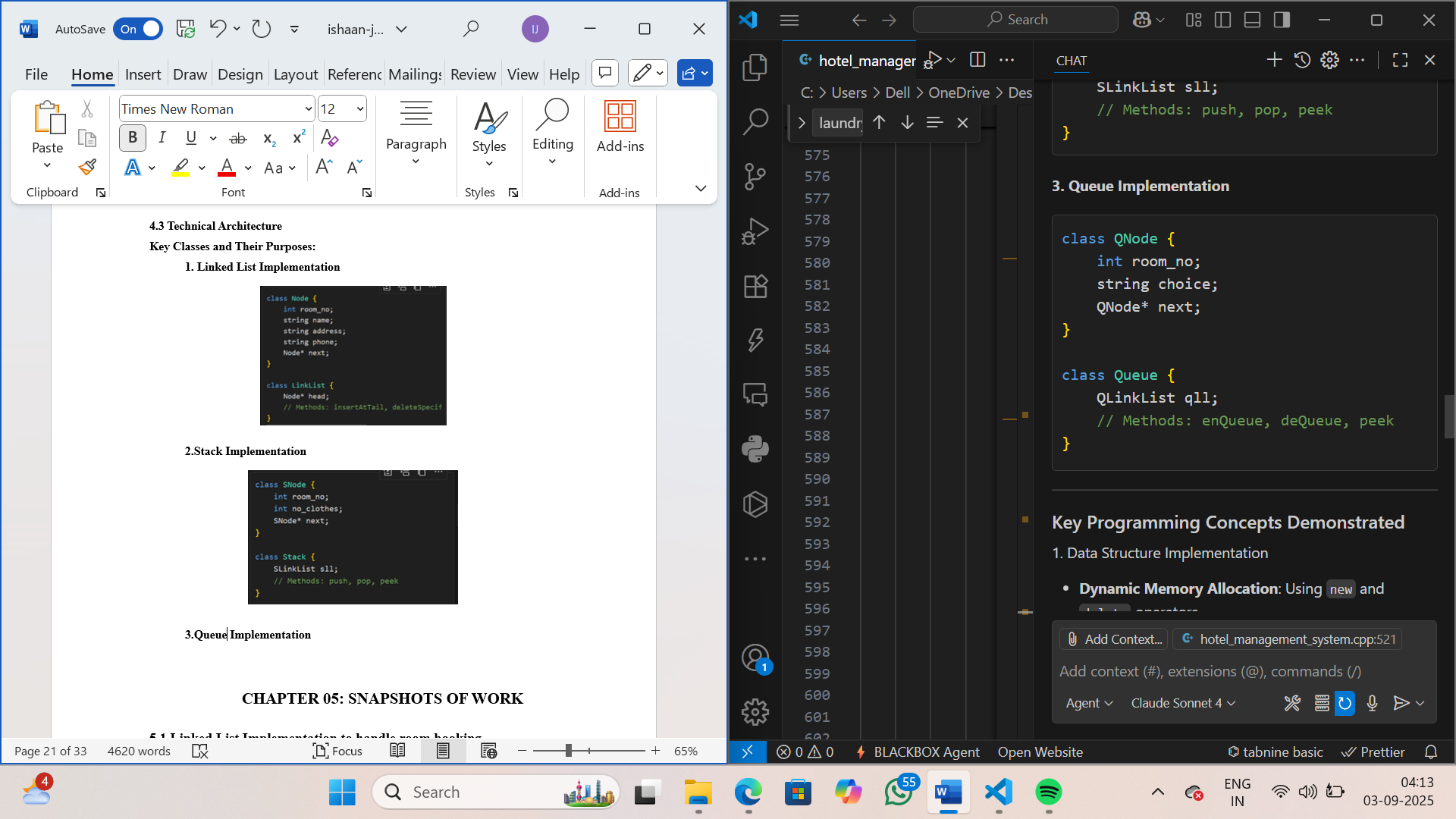
**Key Classes and Their Purposes:**

**1. Linked List Implementation**



**2.Stack Implementation**



**3.Queue Implementation**

**4.4 Key Programming Concepts Demonstrated**

**1. Data Structure Implementation**

* **Dynamic Memory Allocation:** Using new and delete operators
* **Pointer Manipulation:** Linking nodes and traversing structures
* **Memory Management:** Proper cleanup to prevent memory leaks

**2. Object-Oriented Programming**

* **Encapsulation:** Data and methods bundled in classes
* **Abstraction:** Complex operations hidden behind simple interfaces
* **Modularity:** Separate classes for different functionalities

**3. Algorithm Implementation**

* **Insertion Algorithms:** At head, tail, and specific positions
* **Deletion Algorithms:** From head, tail, and specific nodes
* **Traversal Algorithms:** For displaying and searching data

**4.5 System Constraints and Validations**

**1. Room Number Validation**

* **Range**: 1-200 rooms maximum
* **Purpose**: Realistic hotel capacity simulation

**2. Laundry Quantity Limits**

* **Maximum**: 50 clothes per order
* **Purpose**: Practical service limitations

**3. User Interface**

* **Menu-Driven**: Clear navigation between different services
* **Error Handling**: Graceful handling of invalid inputs

**4.6 System Constraints and Validations**

**1. Real-World Simulation**

* **Hotel Industry**: Applicable to actual hotel management scenarios
* **Service Coordination**: Demonstrates systematic service management
* **Customer Experience**: Shows how data structures improve efficiency

**2. Educational Value**

* **Concept Reinforcement**: Practical application of DSA concepts
* **Programming Skills**: Enhancement of C++ programming abilities
* **Problem-Solving**: Development of analytical thinking

**4.7 Future Enhancement Possibilities**

**1. Data Persistence**

* **File I/O**: Save and load data from files
* **Database Integration**: Connect with database systems

**2. Advanced Features**

* **Search Functionality**: Find guests by name or phone
* **Billing System**: Calculate charges for services
* **Reporting**: Generate usage statistics

**3. Additional Services**

* **Housekeeping**: Room cleaning management
* **Event Management**: Conference room bookings

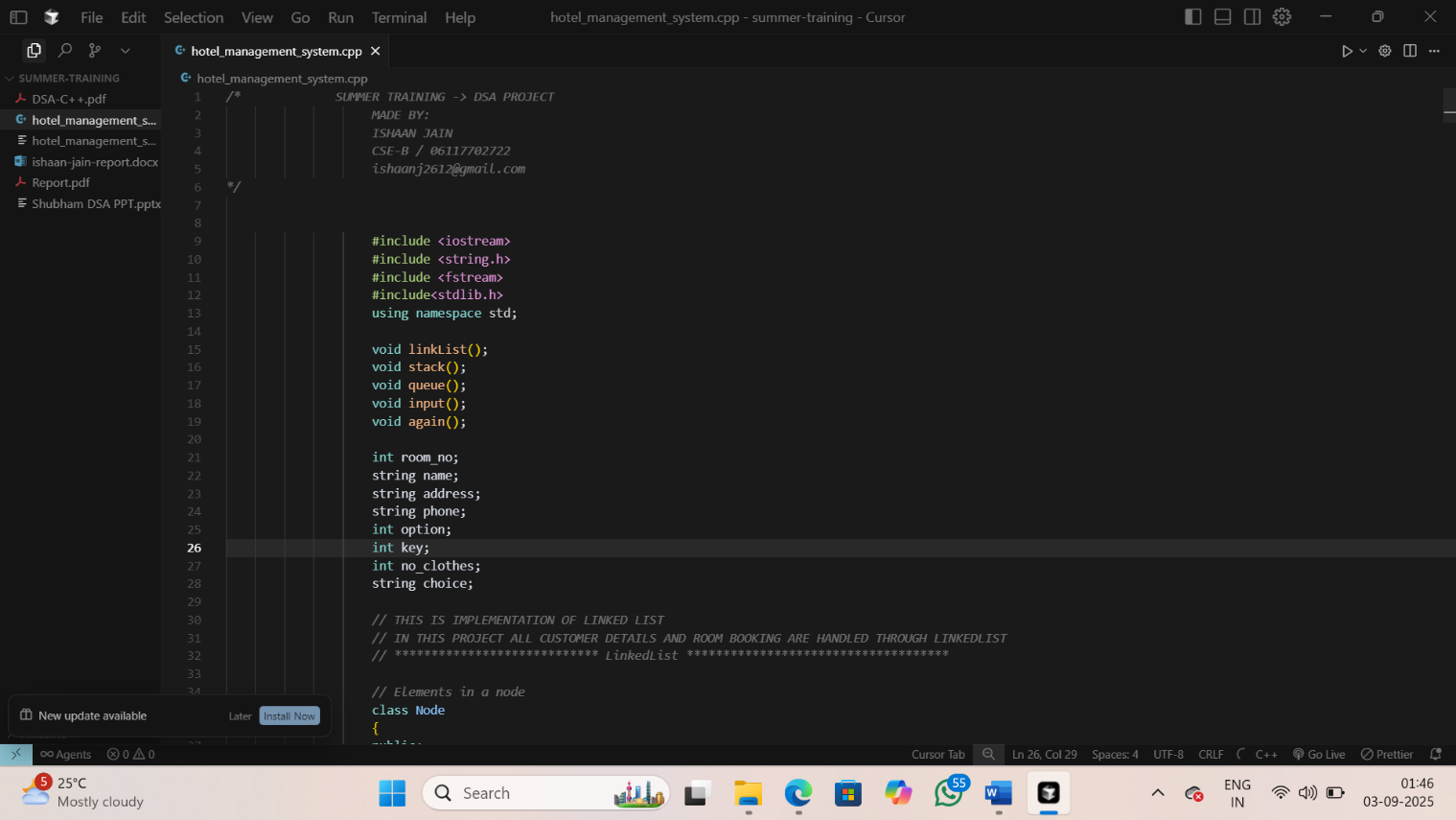
**4.8 Conclusion**

The Hotel Management System successfully demonstrates the practical implementation of fundamental data structures in a real-world business context. The project effectively showcases how Linked Lists, Stacks, and Queues can be utilized to solve different operational challenges in hotel management.

This system serves as an excellent educational tool for understanding data structures while providing a foundation for more complex hotel management solutions. The modular design makes it easy to understand, maintain, and extend for future requirements.

# **CHAPTER 05: SNAPSHOTS OF WORK**

**5.1 Linked List Implementation to handle room booking**



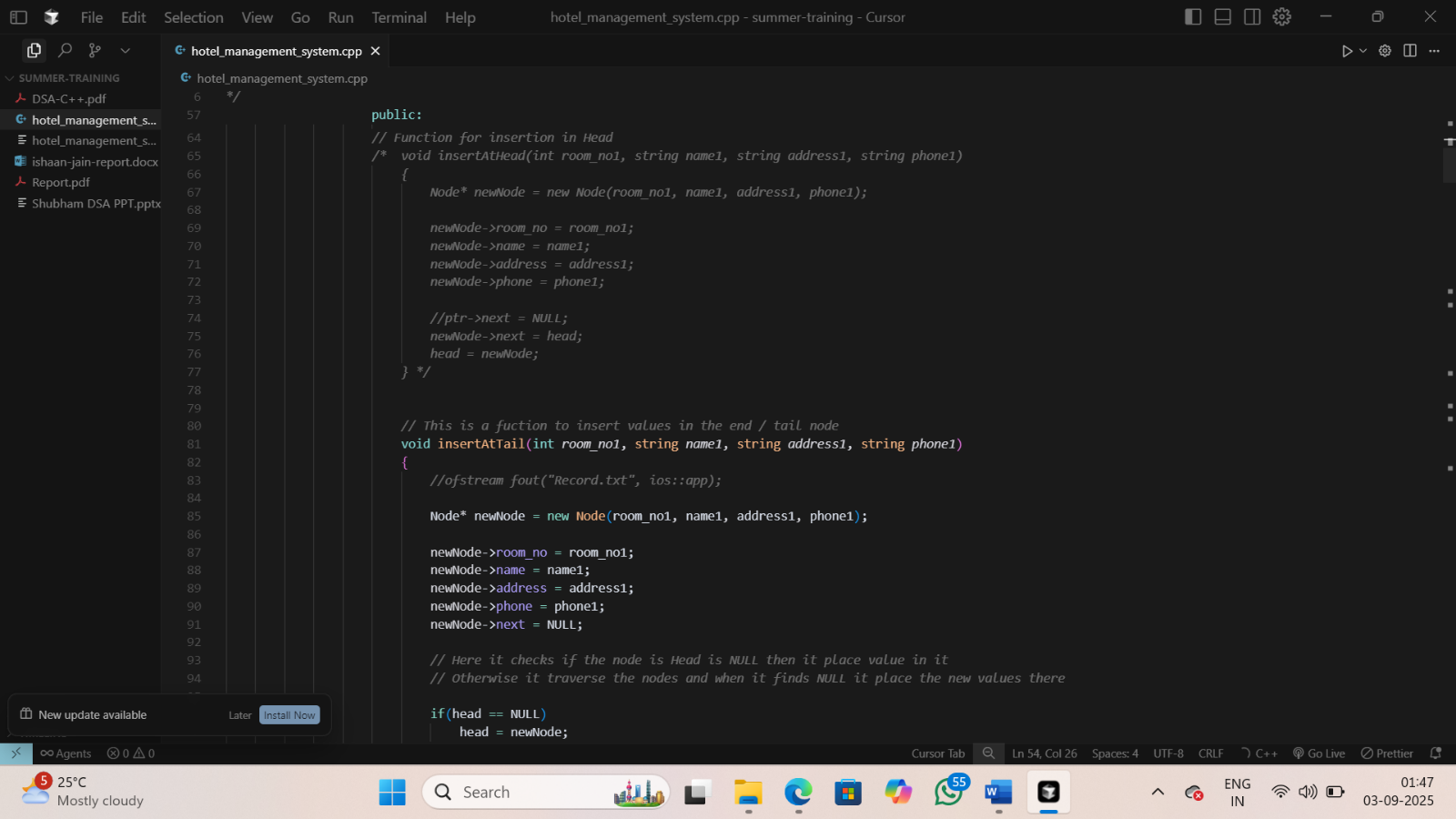
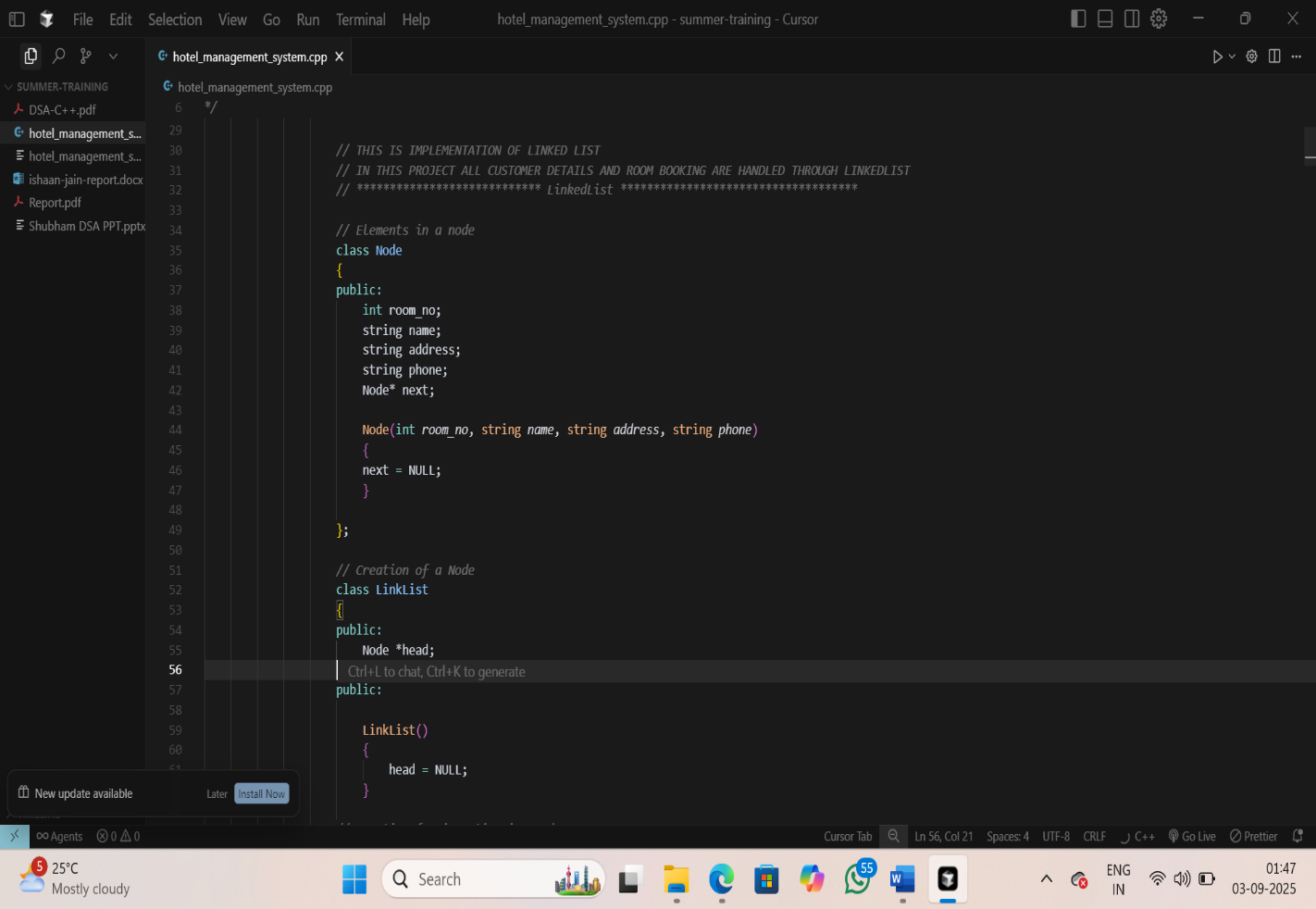
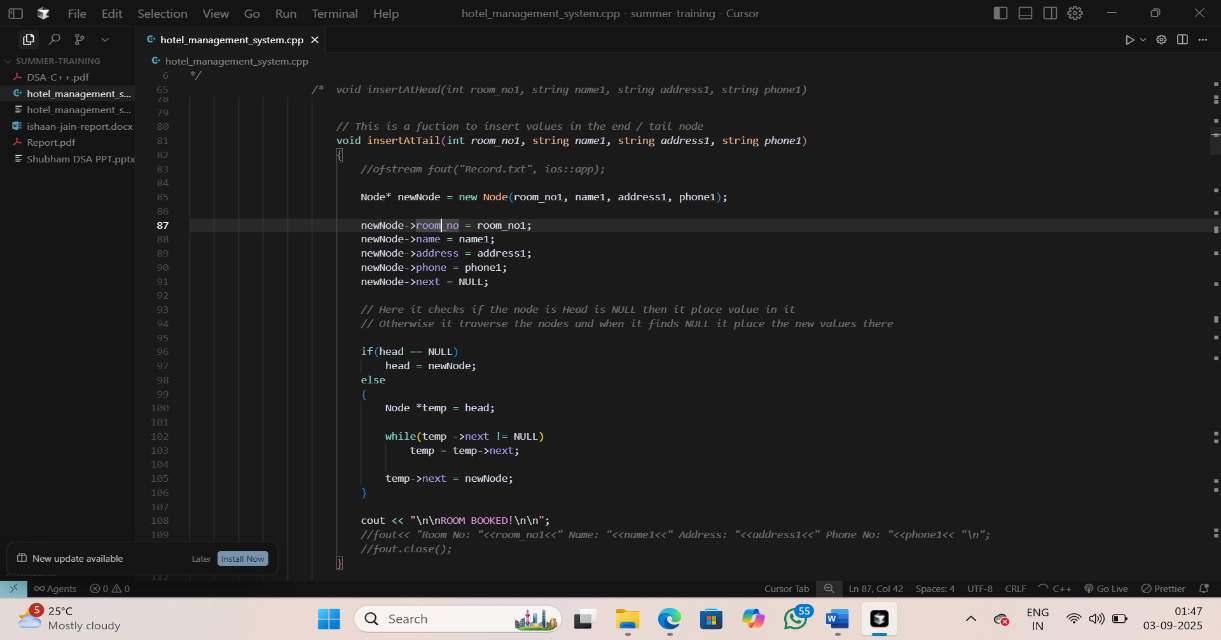
 Figure 5.1.1: Setting up header files and state variables for user management

Figure 5.1.2: Setting up linked list for Customer Details and Room Booking



/

Figure 5.1.3: Function to book the room



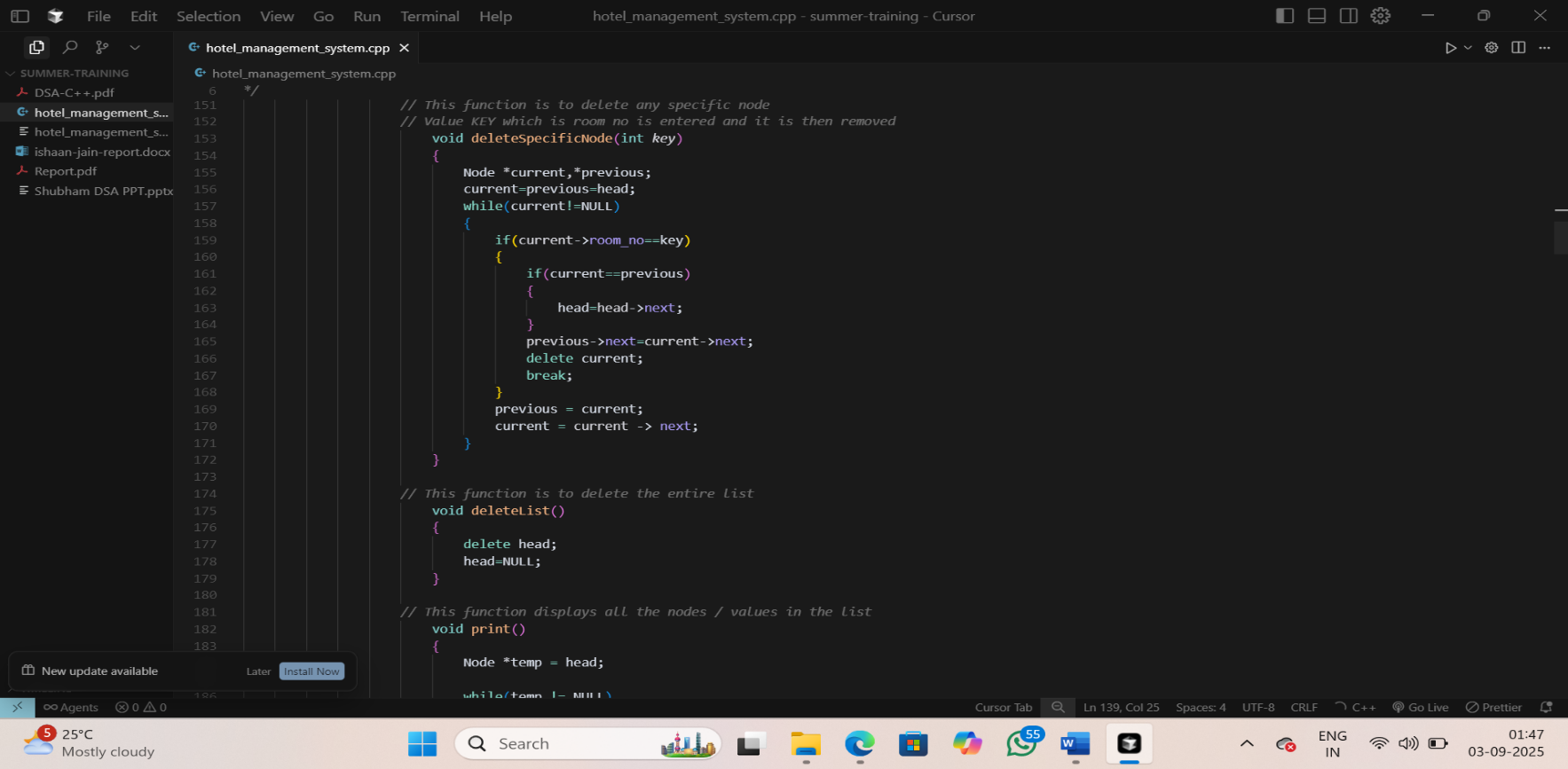


Figure 5.1.4: Function to delete the last booking or specific booking

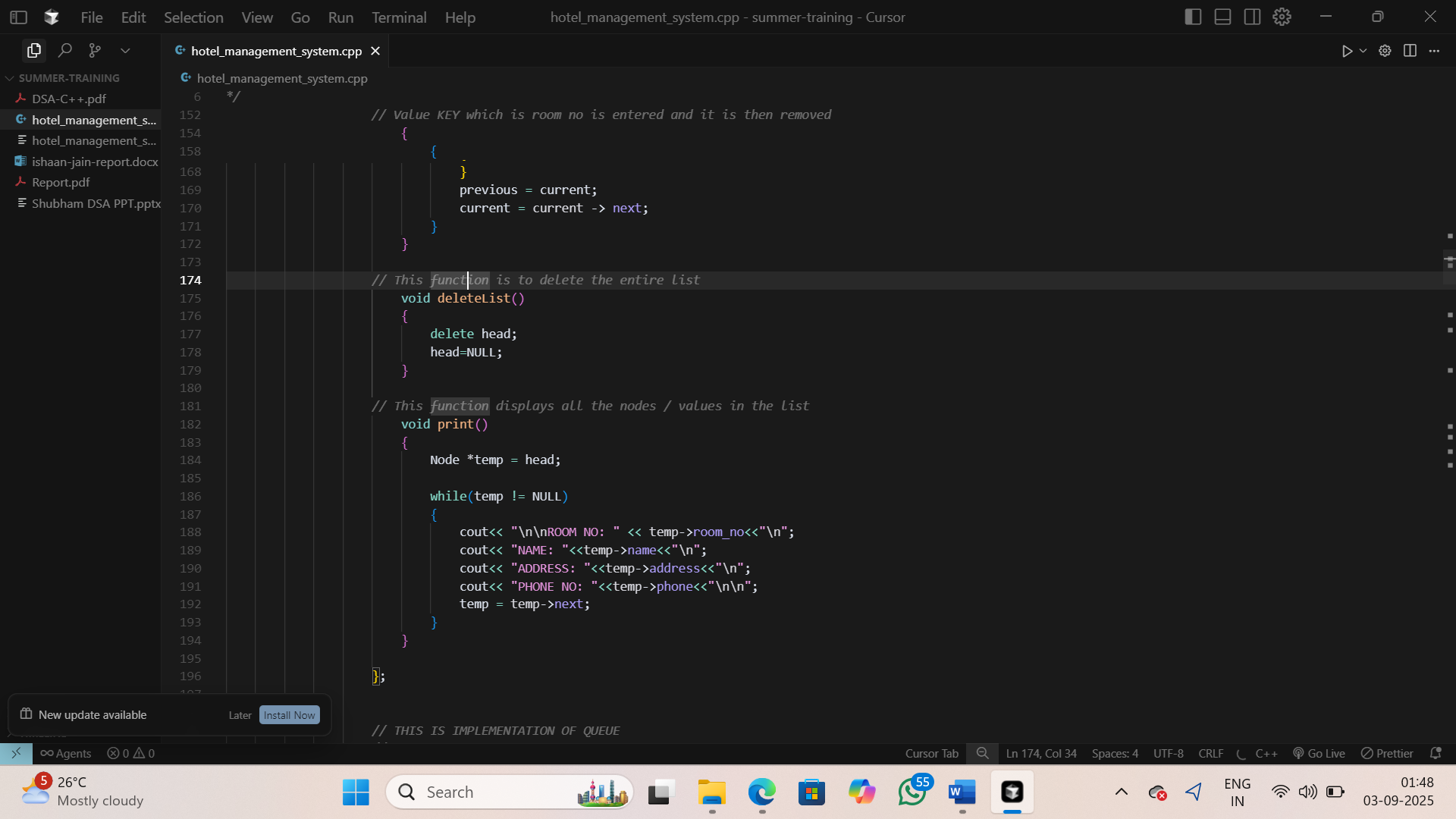


Figure 5.1.5: Function to display all the values and delete all the list

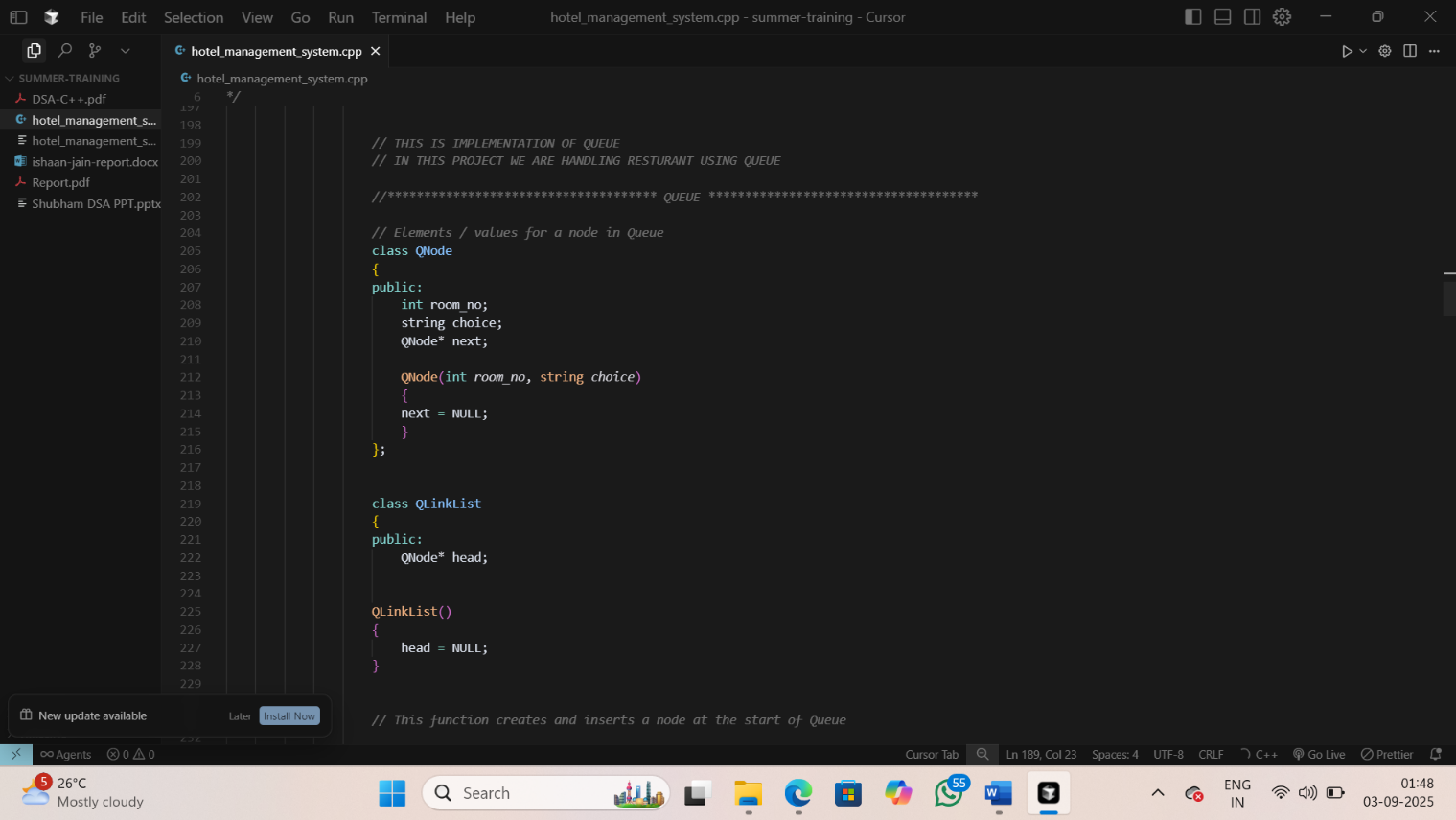
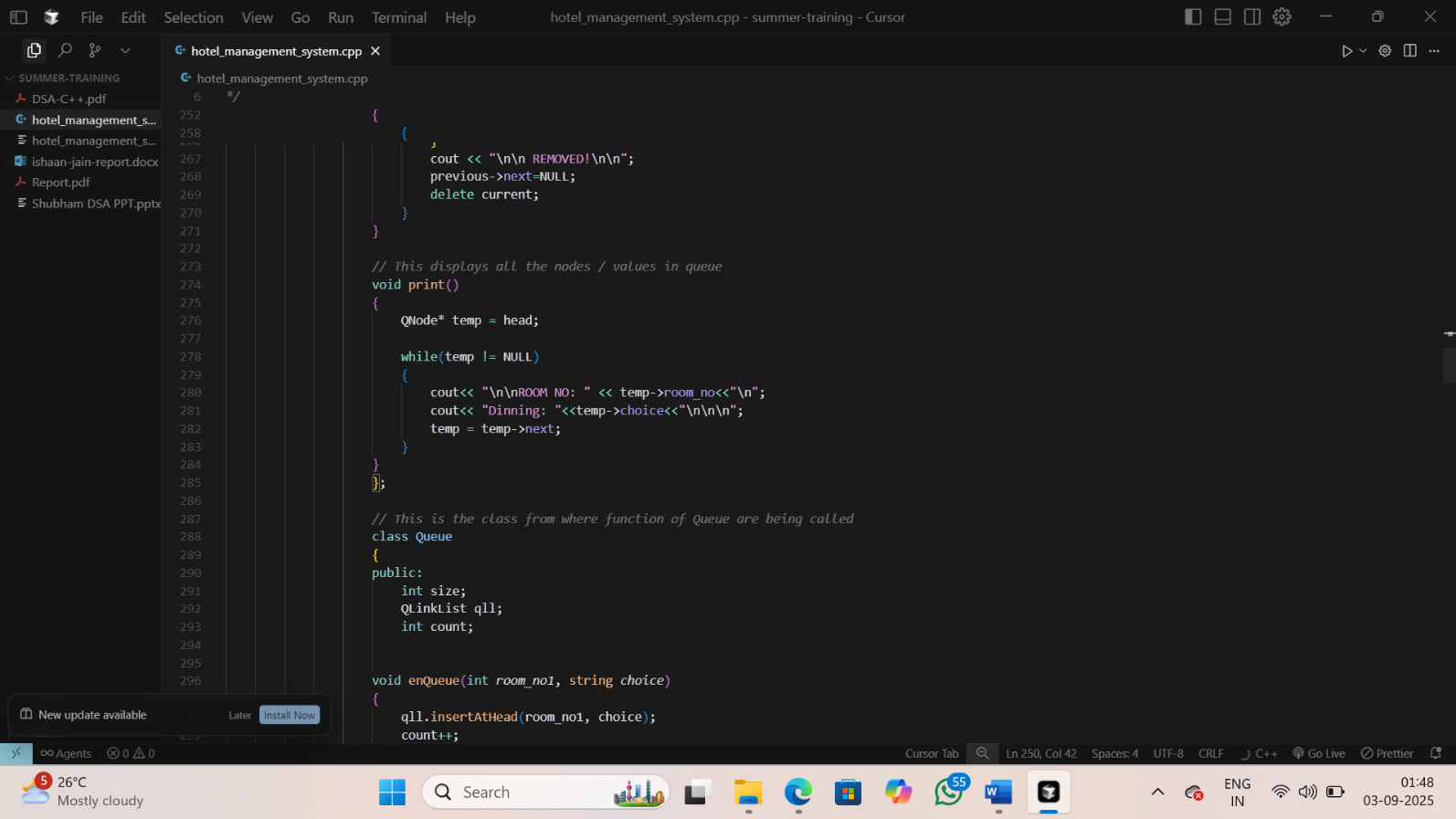
**5.2 Queue Implementation to handle Restaurant**

Figure 5.2.1: Setting up Queue for handling Restaurants



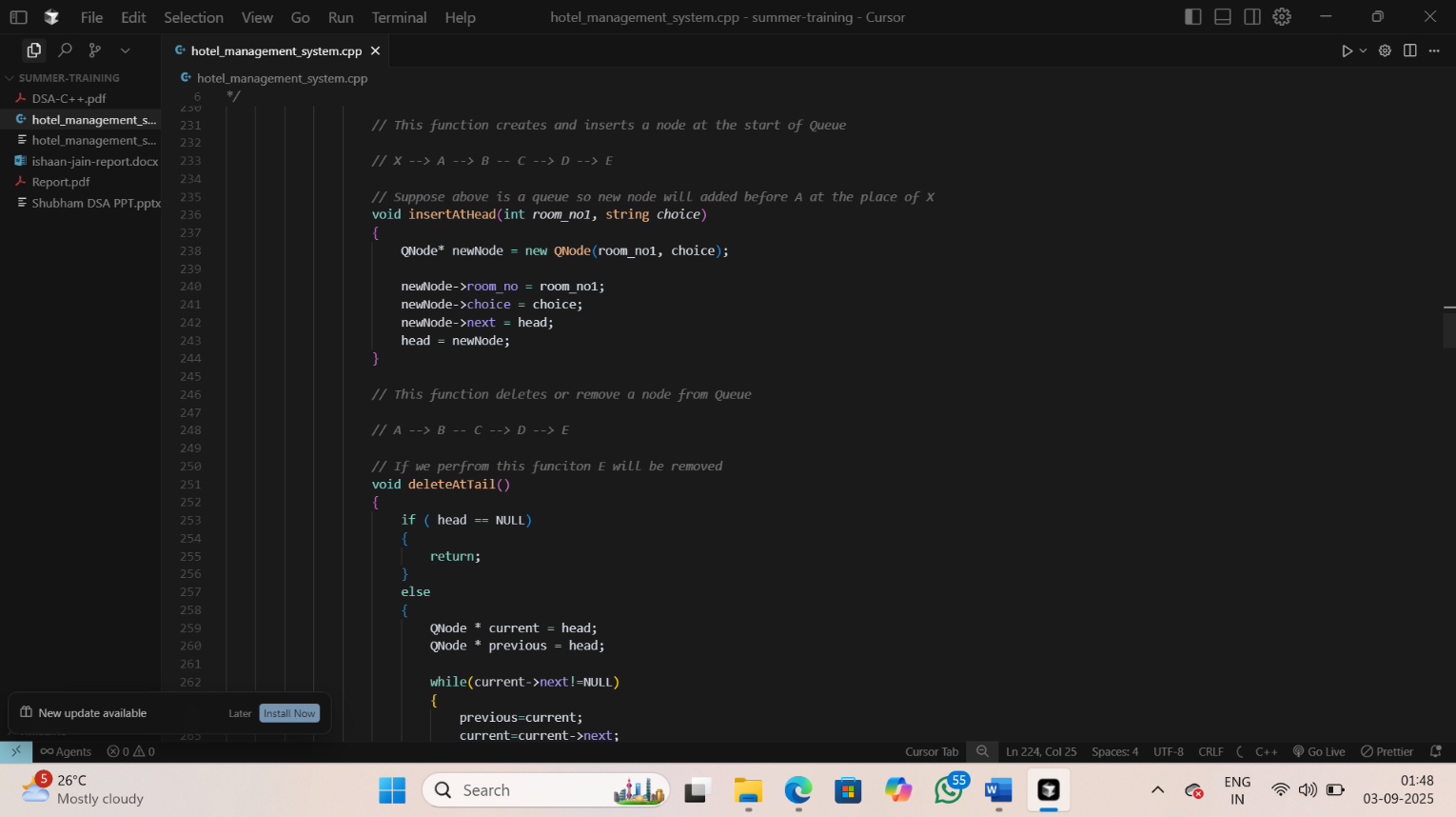


Figure 5.2.2: Function to add at the start, delete from the end, display all the choices of the dinning

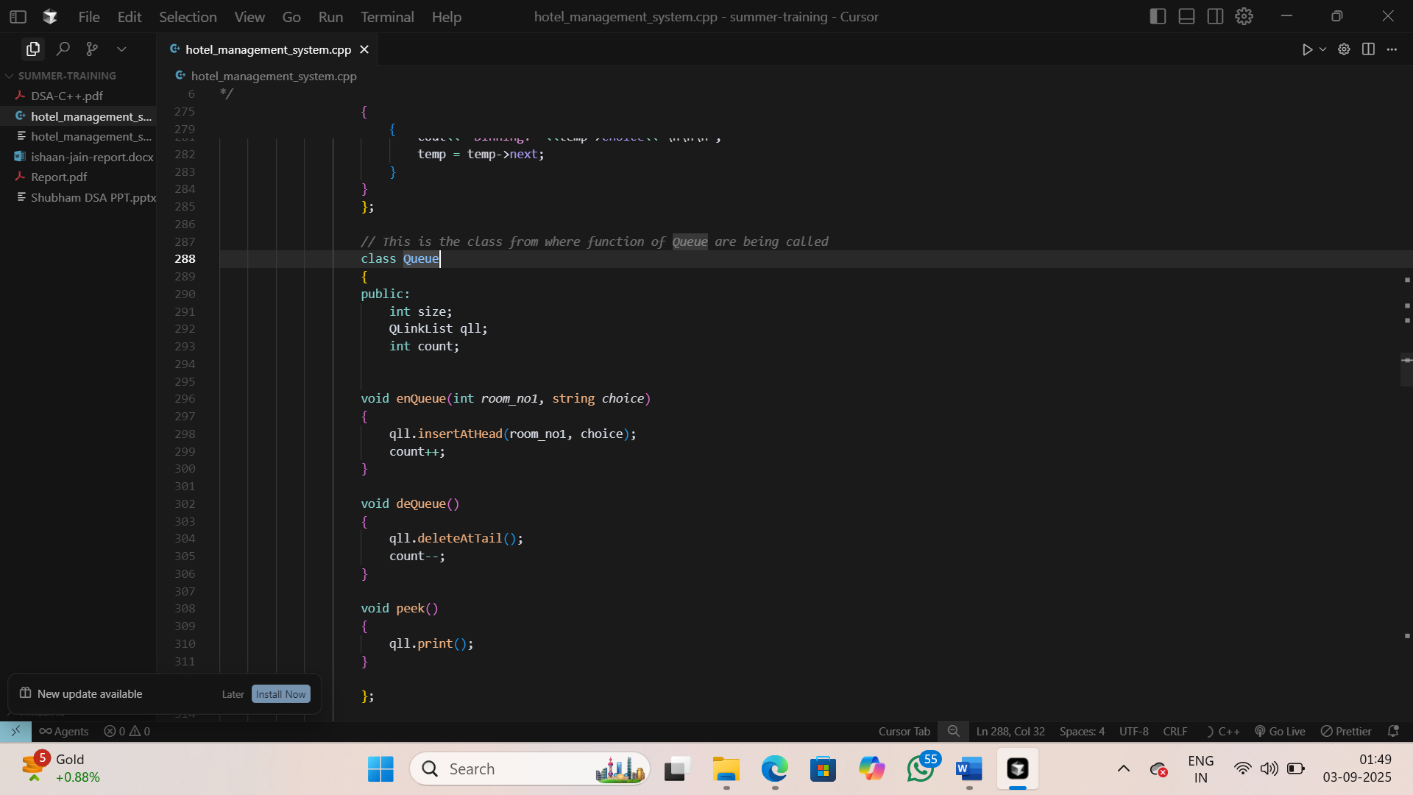
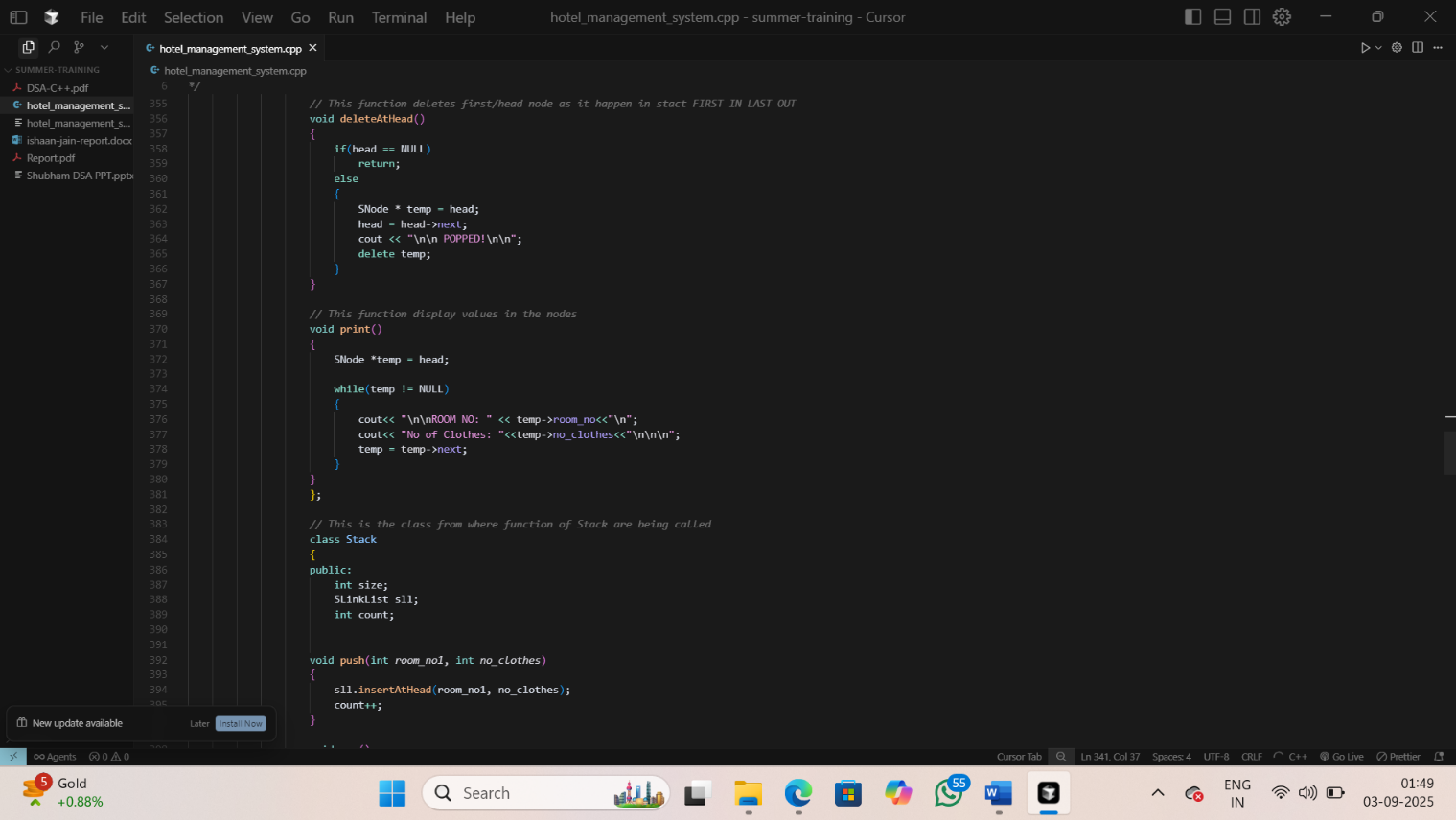


Figure 5.2.3: Queue Class Setup Figure 5.2.4: Function to delete the head and print all the details

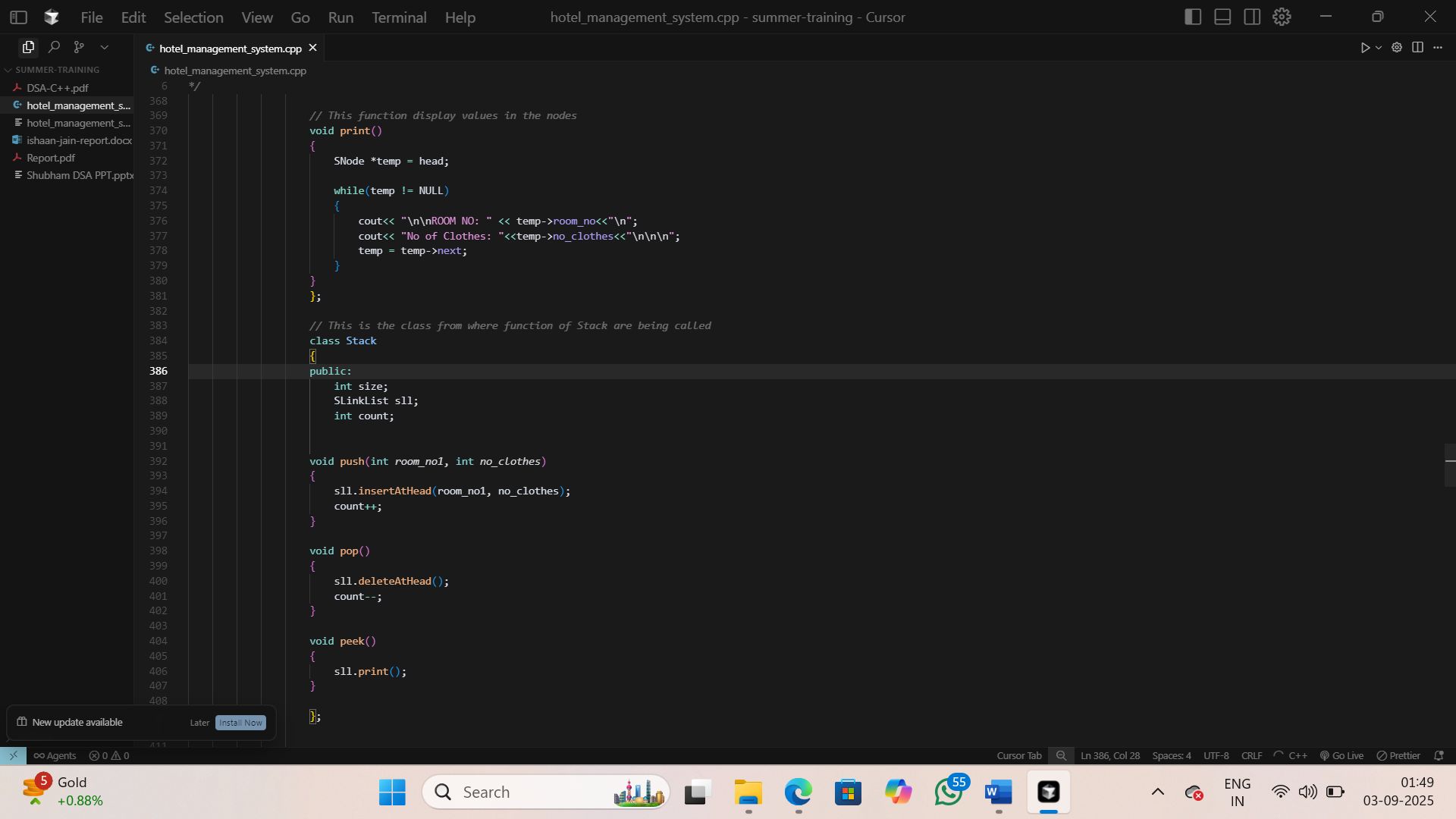
**5.3 Stack Implementation** **for laundry service**

Figure 5.3.1: Stack Class and its pop, push and peek functions

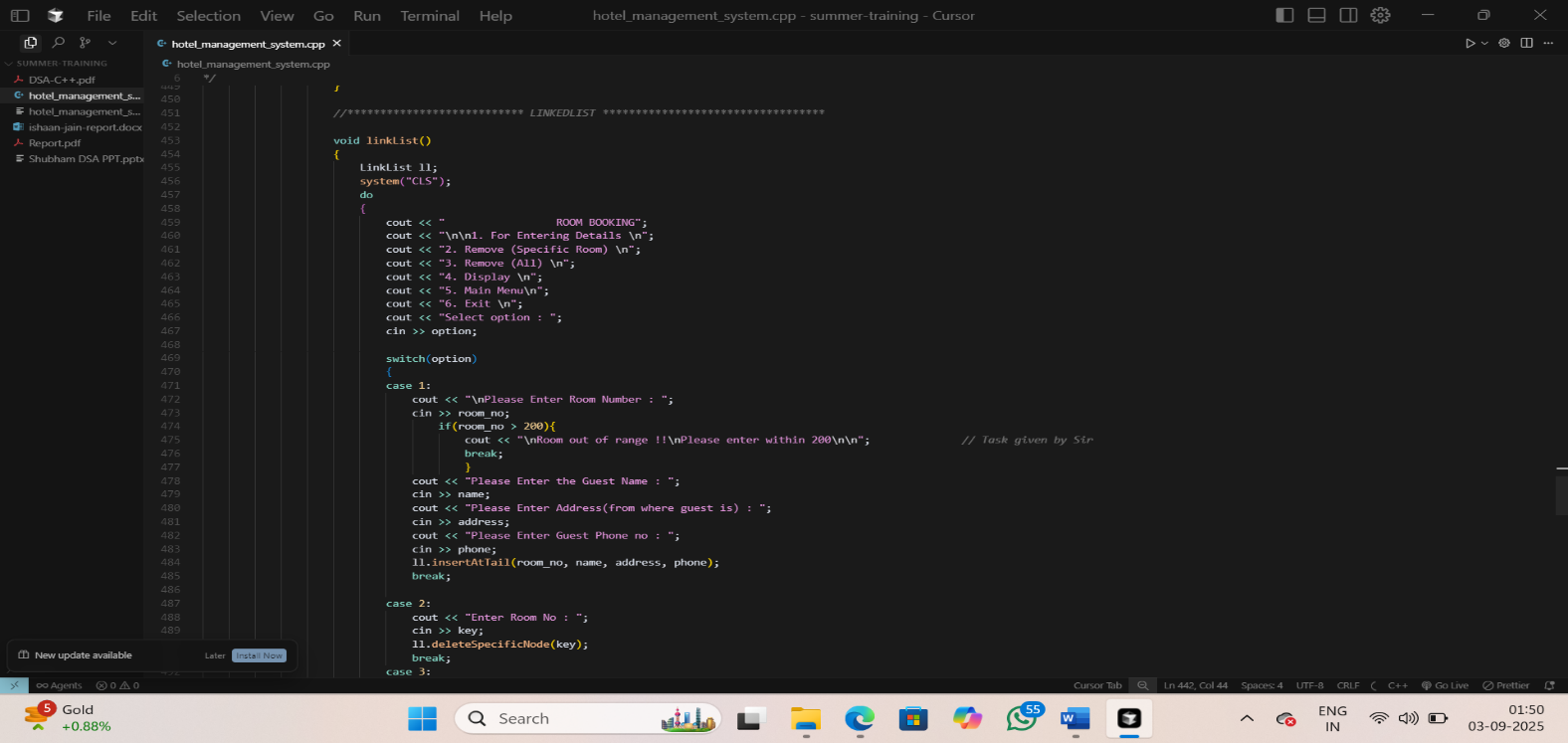
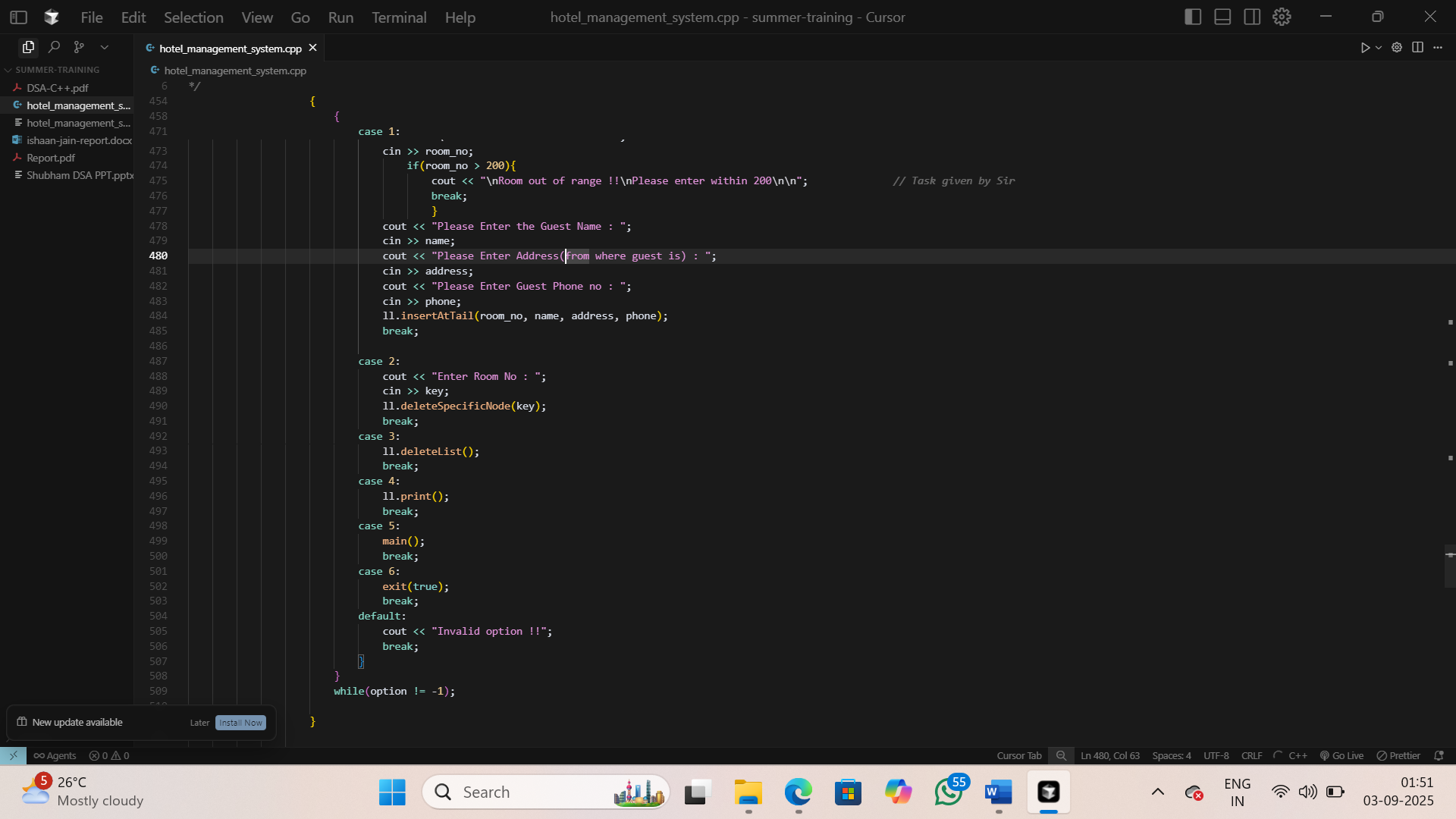
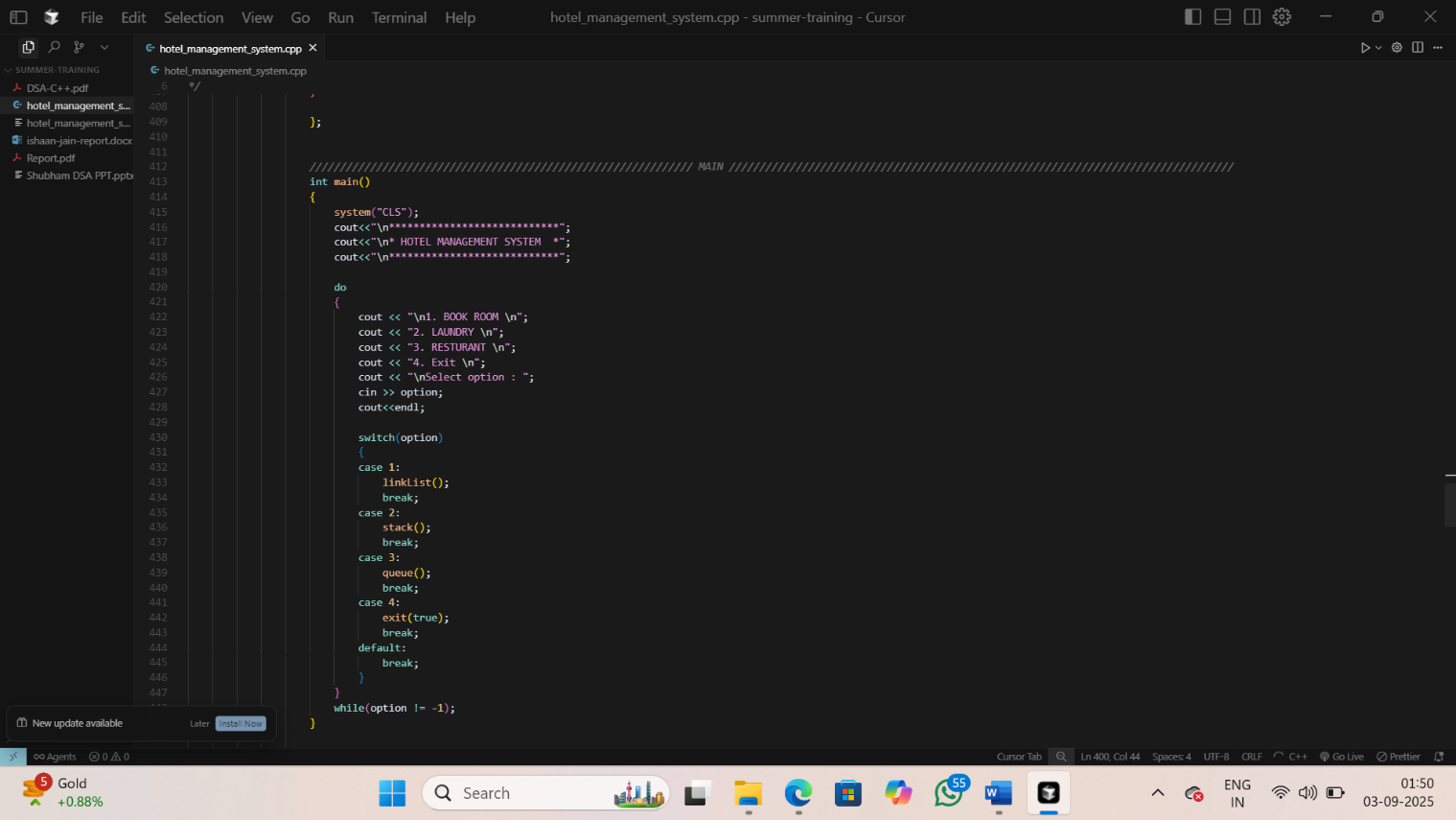
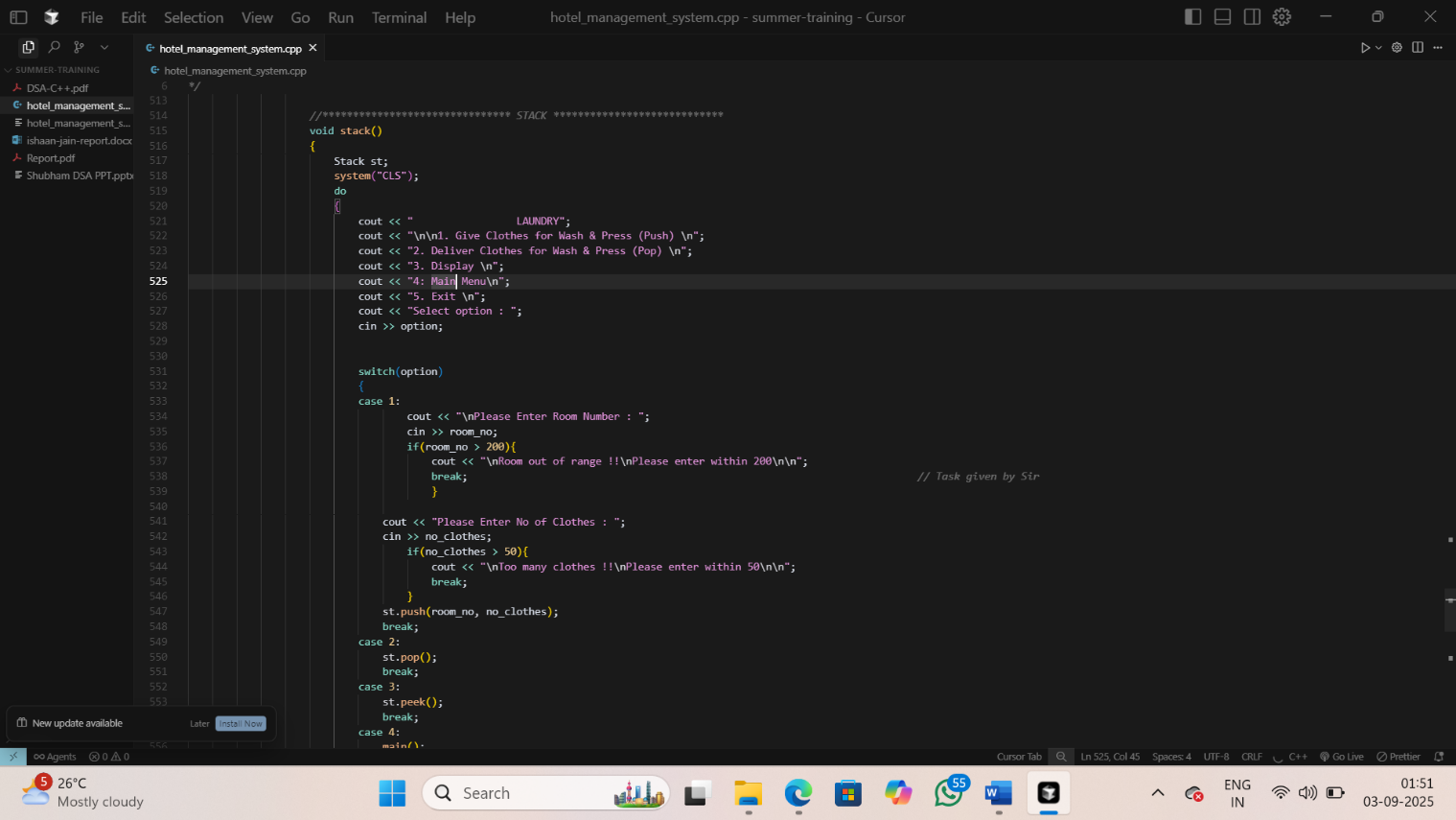
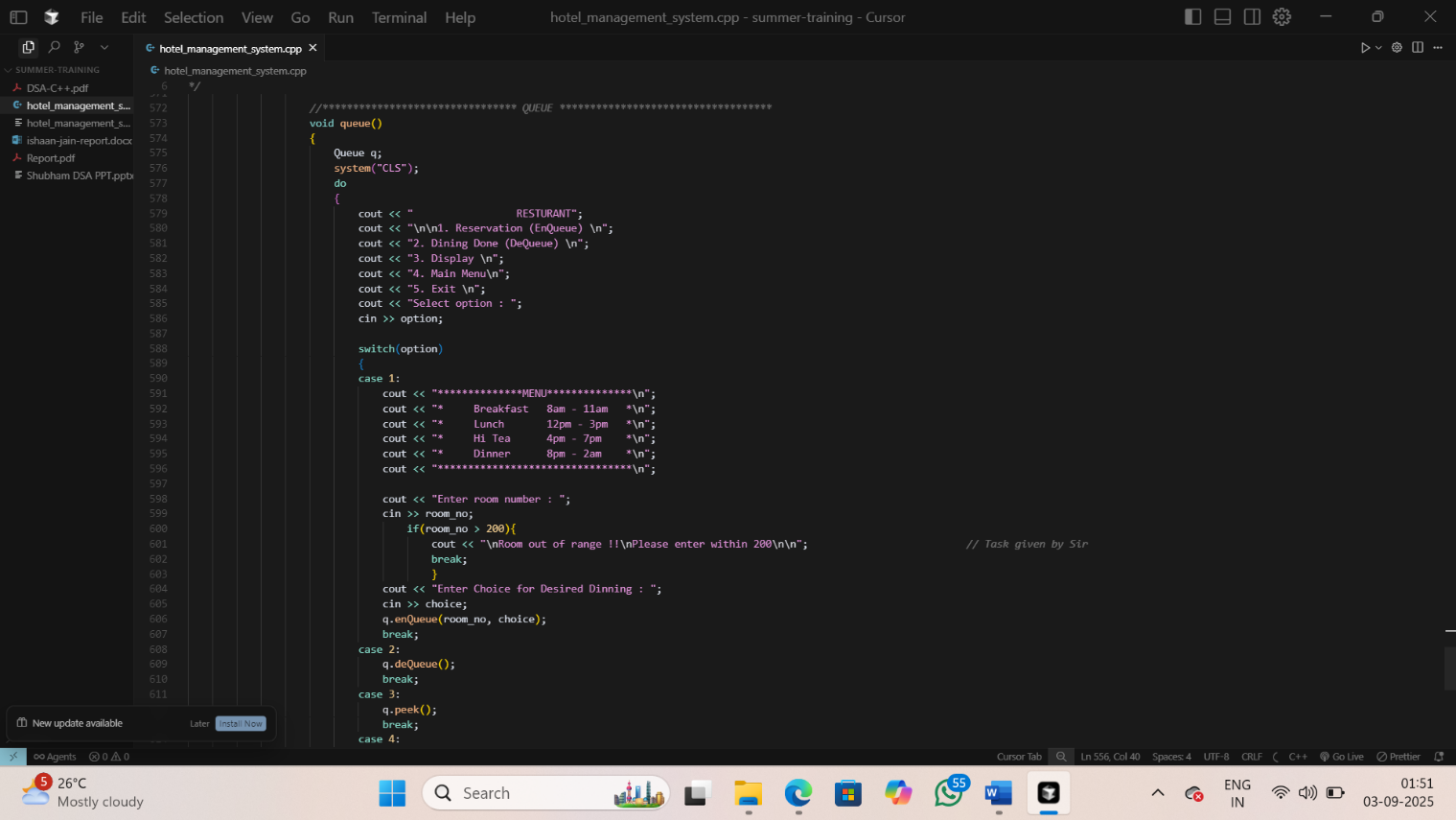
* 1.  **Main Functionalities**

Figure 5.4.1: Choice Selection Figure 5.4.2: Room Booking Selection (laundry, restaurant, book room) (enter details, remove, display bookings)



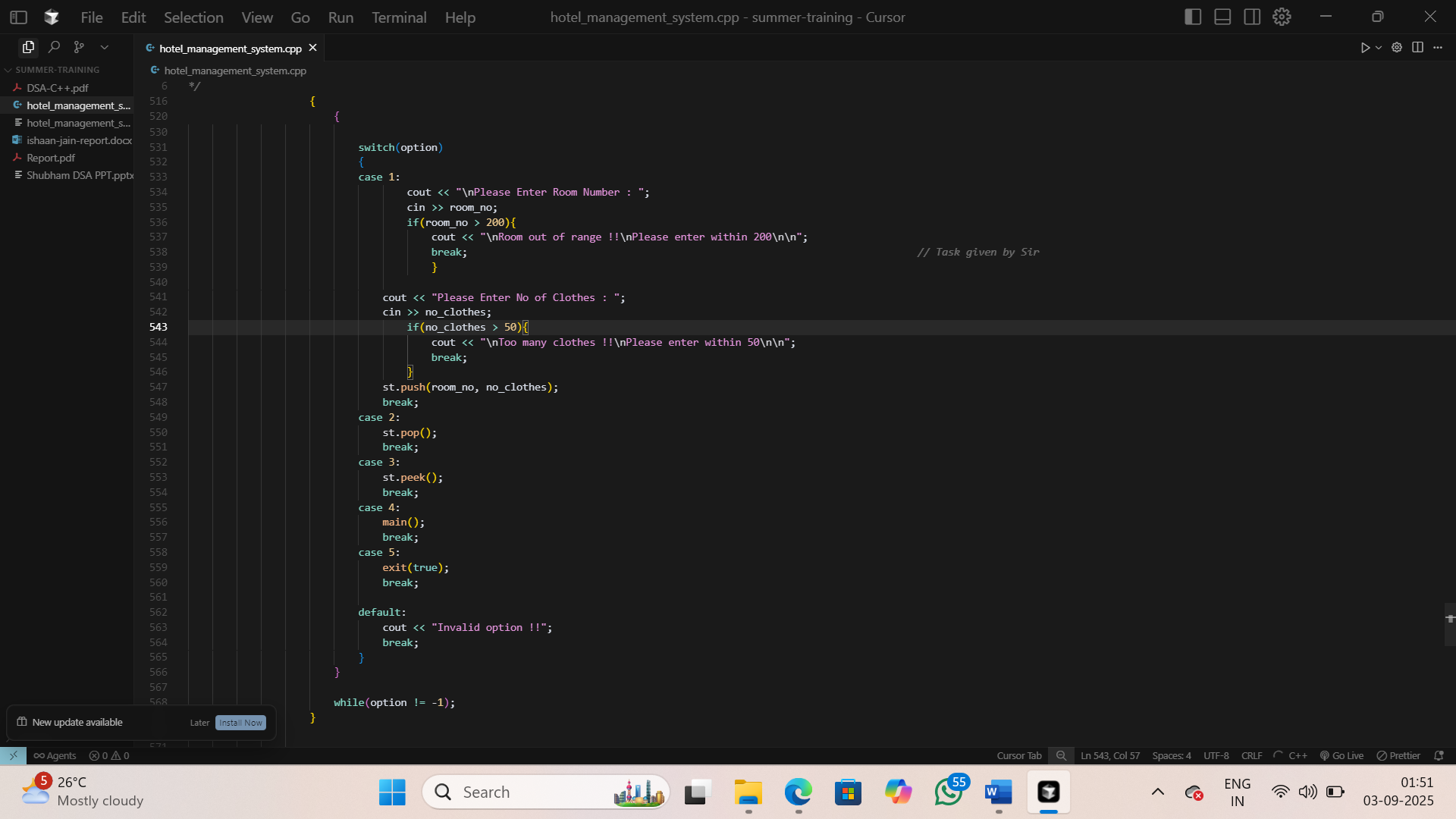


Figure 5.4.4: Restaurant Booking Selection

(reservation, dining, display restaurant booking)

Figure 5.4.3: Laundry Selection

(wash, deliver, display)

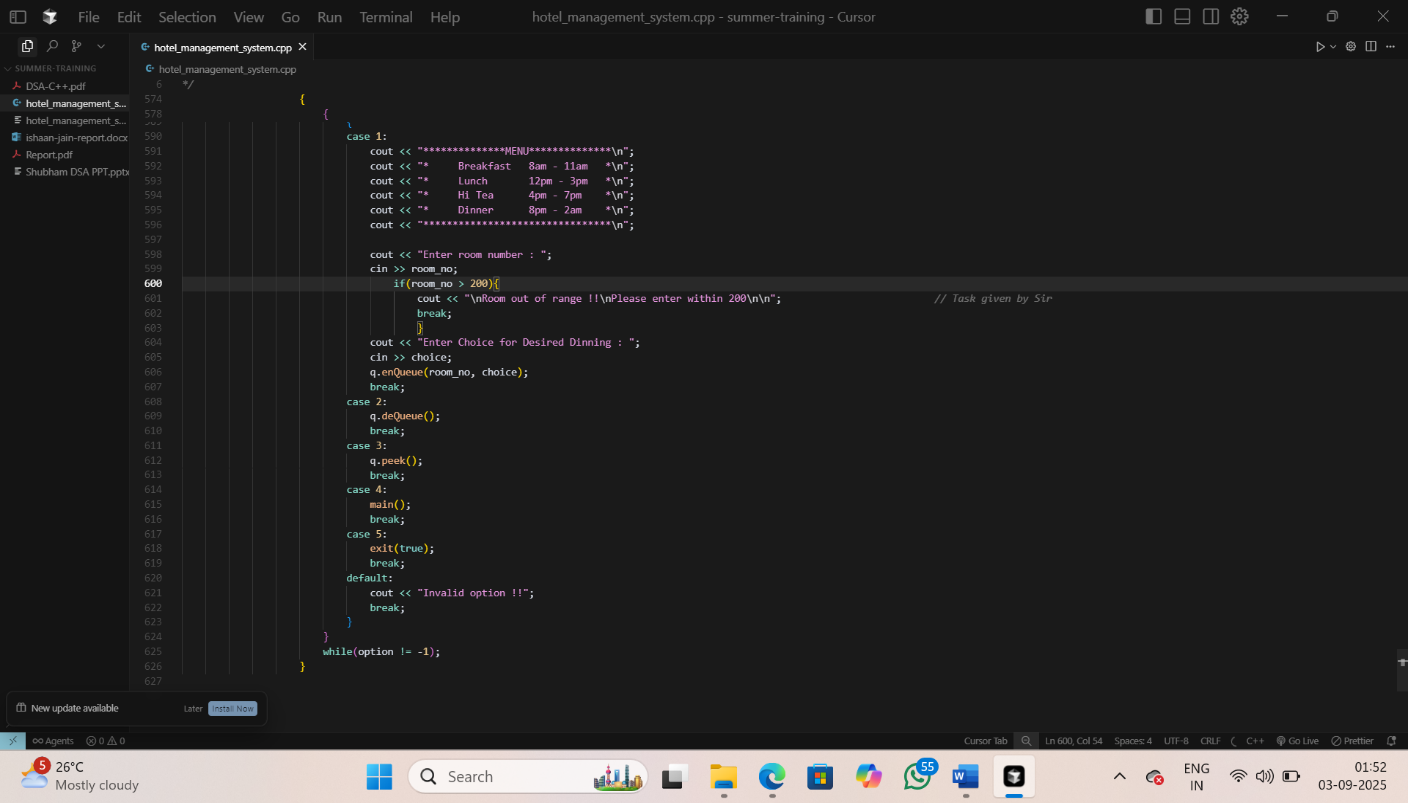


Figure 5.4.5: Stack function call

# **CHAPTER 06: CONCLUSION AND FUTURE SCOPE**

## **6.1 Conclusion of the Project**

This training project on **Data Structures and Algorithms (DSA)** successfully achieved its objective of strengthening problem-solving skills and applying theoretical concepts to practical coding challenges. Through this project, a wide range of data structures (arrays, linked lists, stacks, queues, trees, graphs, hash maps, etc.) and algorithms (searching, sorting, recursion, dynamic programming, greedy methods, and graph algorithms) were studied, implemented, and analyzed.

The project provided hands-on experience with coding platforms like **Coding Blocks IDE** and other online judges, where problems were solved under constraints of time and space complexity. This not only improved algorithmic thinking but also helped in understanding how to optimize solutions for real-world applications.

Overall, the project served as a solid foundation in computational logic, efficient coding practices, and competitive programming readiness.

## **6.2 Key Learning Outcomes**

This DSA training project was a significant learning journey, providing both technical and personal growth.

The key outcomes are categorized below:

* **Technical Skills:** 
  + **Data Structures Mastery:** Gained clarity in implementing and applying arrays, linked lists, stacks, queues, trees, heaps, graphs, and hash tables.
  + **Algorithmic Thinking:** Learned key algorithmic paradigms such as divide and conquer, dynamic programming, greedy strategies, and backtracking.
  + **Complexity Analysis:** Strengthened understanding of **time and space complexity (Big O, Big Θ, Big Ω)** to evaluate the efficiency of algorithms.
  + **Problem-Solving:** Practiced solving problems across difficulty levels to improve coding speed and accuracy.
  + **Programming Practice:** Enhanced coding skills in C++/Java/Python (as per training language), focusing on clean, optimized, and modular code.
* **Soft Skills:** 
  + Improved **logical reasoning and analytical skills** through structured problem-solving.
  + Developed **discipline and consistency** by practicing problems daily on structured tracks.
  + Learned **time management** while attempting problems under contest or exam-like scenarios.
  + Enhanced **debugging and error-resolution** skills by working through runtime and logical errors.

## **6.3 Limitations and Future Enhancements**

While the project fulfilled its main purpose, there is still scope for extending learning and applications:

* **Current Limitations:** 
  + Some advanced topics (e.g., Segment Trees, Fenwick Trees, Advanced Graph Algorithms like Flow Networks) were only touched upon briefly.
  + Real-world implementation of DSA in large-scale system design was beyond the project scope.
  + Competitive programming practice is still limited and requires further speed optimization.
* **Proposed Future Enhancements:** 
  + **Advanced Topics:** Explore advanced data structures (Suffix Arrays, Tries, Disjoint Sets) and algorithms (Network Flows, String Matching).
  + **System Design Integration:** Learn how DSA concepts are applied in scalable software systems and interview problem-solving.
  + **Competitive Programming:** Participate in coding contests (Codeforces, CodeChef, LeetCode) to apply knowledge under time pressure.
  + **Projects using DSA:** Implement real-world projects like a **Pathfinding Visualizer, Recommendation System, or Mini Compiler** to demonstrate DSA in practice.
  + **Collaborative Learning:** Engage in group coding sessions or hackathons for peer-to-peer learning.

## **6.4 Practical Project Highlight: Hotel Management System**

As a culminating practical application of the Data Structures and Algorithms concepts learned, a comprehensive Hotel Management System was developed using C++. This mini-project exemplifies the use of three fundamental data structures, each tasked with solving specific operational challenges in hospitality management through a console-driven menu interface.

* **Room Booking (Linked List):**

This module efficiently handles guest reservations and room assignments for up to 200 rooms. Dynamic insertion and deletion operations facilitate booking new rooms, canceling existing bookings, and checking availability. The linked list’s flexibility allows smooth update of customer and room data with minimal overhead.

* **Laundry Service (Stack - LIFO):**

The laundry module simulates the collection and delivery of clothes using a stack data structure, ensuring the last received laundry requests are processed first. It incorporates constraints such as a maximum of 50 clothes per request and 200 rooms served, modeling a fair and practical processing order.

* **Restaurant Service (Queue - FIFO):**

Managing dining reservations through a queue guarantees a first-come, first-served fairness principle. This service supports time-segmented meals (Breakfast, Lunch, Hi Tea, Dinner) for up to 200 rooms, enabling orderly reservation and servicing of guests.

The implementation employs dynamic memory management (new and delete) along with a class-based design to ensure modularity and clarity. Input validation and error handling mechanisms uphold robustness. This project vividly demonstrates how selecting the appropriate data structure — linked list, stack, or queue — can effectively address distinct real-world problems in the hospitality domain.

## **6.5 Integration with Learning and Future Enhancements**

This project enriched the overall training experience by offering hands-on exposure to applying data structures in practical scenarios involving scheduling, resource management, and user interactions. It strengthened skills in designing modular, efficient code while deepening understanding of algorithmic problem-solving tailored to specific operational needs (e.g., LIFO for laundry fairness, FIFO for dining reservations).

Potential avenues for future work include:

* Introducing advanced data structures such as balanced trees or hash maps for faster data lookup and management.
* Adding a graphical user interface (GUI) to enhance user experience and accessibility.
* Extending functionality with billing systems, analytical reporting, and concurrency control for multi-user environments.
* Incorporating database integration for persistent data storage and retrieval.

# **CHAPTER 07: REFERENCES**

* Coding Blocks: *Data Structures and Algorithms Course Material* – https://codingblocks.com/
* GeeksforGeeks: *Data Structures and Algorithms Tutorials* – https://www.geeksforgeeks.org/
* CLRS: *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein.
* HackerRank / LeetCode: *Online Practice Platforms for DSA*.
* Abdul Bari: *YouTube Lectures on Algorithms and Data Structures*.