Annexure-I

Basics of Data Structures and Algorithms – PETV70

CENTRE FOR PROFESSIONAL ENHANCEMENT (LPU)

A training report

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

B.TECH in COMPUTER SCIENCE & ENGINEERING

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 06/10/25 to 07/20/25

SUBMITTED BY

Name of student: Ambika Raj

Registration Number: 12311532

Signature of the student:

Ambika

Annexure-II: Student Declaration

To whom so ever it may concern

I, Ambika Raj, 12311532, hereby declare that the work done by me on "Basics of Data Structures and Algorithms: Tic-Tac-Toe Game Project" from June, 2025 to July, 2025, is a record of original work for the partial fulfillment of the requirements for the award of the degree, B.TECH IN COMPUTER SCIENCE & ENGINEERING.

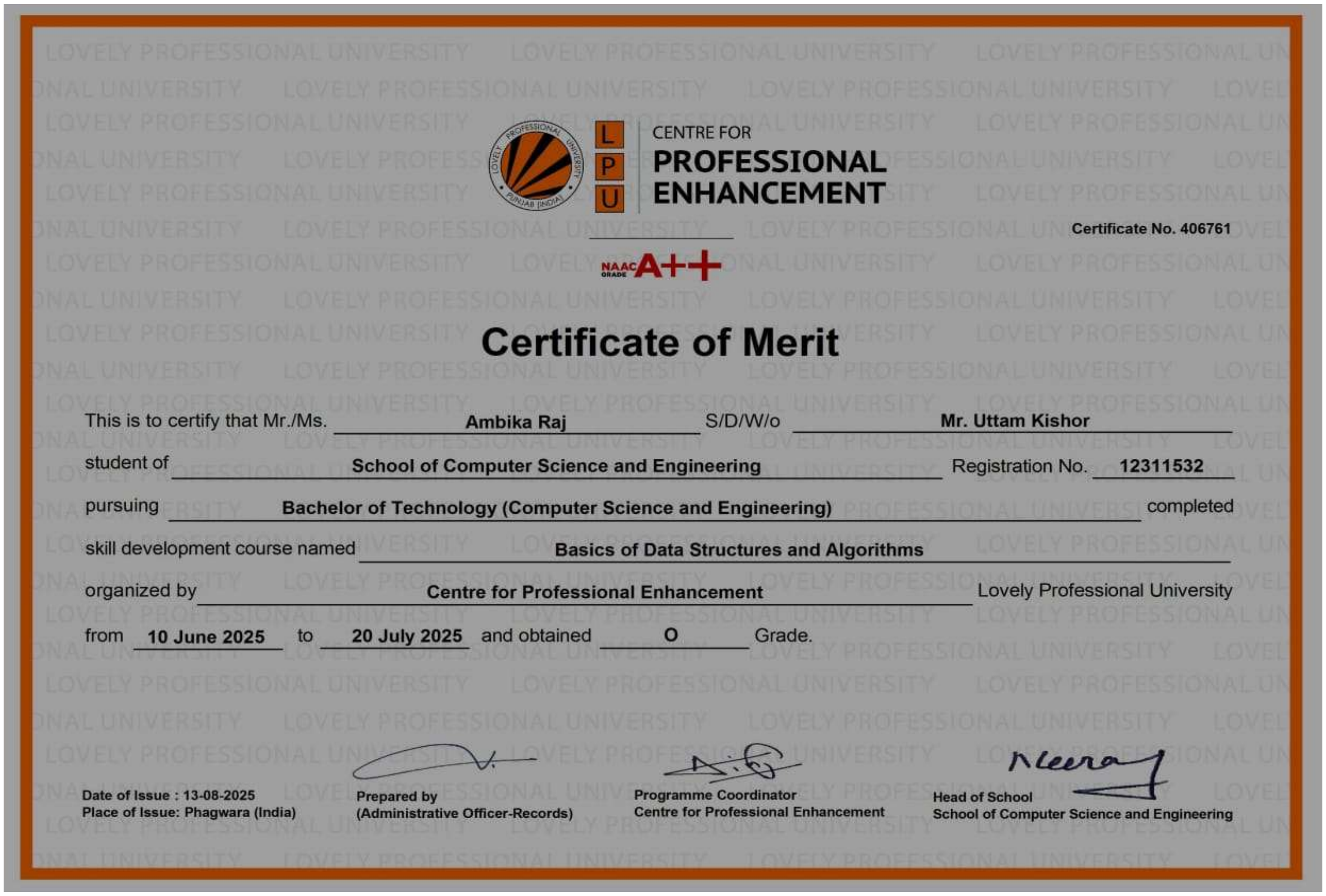
Ambika Raj (12311532)

Signature of the student

Ambika

Dated: 31 August,2025

3.Training Certification from organization:-



CENTRE FOR PROFESSIONAL ENHANCEMENT

LPU

NAAC GRADE A++

Certificate No. 406761

Certificate of Merit

This is to certify that Mr./Ms. **Ambika Raj** S/D/W/o **Mr. Uttam Kishor**

student of **School of Computer Science and Engineering** Registration No. **12311532**

pursuing **Bachelor of Technology (Computer Science and Engineering)** completed

skill development course named **Basics of Data Structures and Algorithms**

organized by **Centre for Professional Enhancement** Lovely Professional University

from **10 June 2025** to **20 July 2025** and obtained **O** Grade.

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

[illegible]
Prepared by
(Administrative Officer-Records)

[illegible]
Programme Coordinator
Centre for Professional Enhancement

Neeraj
Head of School
School of Computer Science and Engineering

4.Acknowledgement:-

I would like to express my sincere gratitude to **Lovely Professional University (LPU)** for providing me the opportunity to undergo the **Summer Training Program** in the subject *Data Structures and Algorithms (DSA)*, which has immensely helped me bridge the gap between theoretical knowledge and practical implementation.

First and foremost, I am deeply thankful to my mentors, **Ms. Sudha Shanker Prasad**, for their constant guidance, valuable feedback, and encouragement throughout the training and the project work. Their insights into the subject have not only enhanced my technical knowledge but also developed my confidence in problem-solving and logical thinking.

I would also like to extend my heartfelt thanks to the **School of Computer Science and Engineering (CSE)**, LPU, for designing such a structured and practical-oriented training program. The training has given me the opportunity to work on real-world problems, understand the applications of **arrays, stacks, queues, linked lists, trees, and graphs**, and finally implement these learnings into a creative project.

Special thanks to my family and friends for their continuous motivation and support during the entire course of this project. Their encouragement kept me focused and determined, especially when I faced challenges in logic building and debugging.

Lastly, I am grateful to my peers who provided constructive discussions and healthy competition throughout the training period. Such collaboration made the learning process more enjoyable and effective.

This acknowledgement would remain incomplete without mentioning the incredible resources provided by **LPU's digital library, online lectures, and training workshops**, which acted as an enabler for me to complete my project successfully.

I humbly acknowledge all those who directly or indirectly contributed to my learning journey.

Name: Ambika Raj

Registration Number: 12311532

5. **LIST OF TABLES**

6. **LIST OF FIGURES**

7. **LIST OF ABBREVIATIONS**

Abbreviation	**Full Form**
DSA	Data Structures and Algorithms
UI	User Interface
UX	User Experience
UML	Unified Modeling Language
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
AI	Artificial Intelligence
ML	Machine Learning

Chapter 1. INTRODUCTION OF THE PROJECT UNDERTAKEN

1.1 Objectives of the Work Undertaken

The primary aim of my summer training project was to **apply the concepts of Data Structures and Algorithms (DSA) in a real-world application**, thereby strengthening my understanding of both theoretical and practical aspects of the subject. While traditional classroom learning familiarizes us with definitions, pseudo-code, and isolated exercises, this training demanded that we **translate theory into practice** by developing a project that demonstrates efficiency, creativity, and functionality.

The following specific objectives were defined and pursued during this project:

1. To Strengthen Understanding of Core Data Structures

One of the foremost objectives of this training was to revisit and apply the fundamental data structures taught in class such as **arrays, stacks, and objects**.

- The **array** was used to represent the 3×3 Tic Tac Toe game board.
- A **stack-like approach** was applied in maintaining move history, enabling the undo functionality.
- **Objects** were used to manage taunts and store temporary state information.

This practical integration of structures ensured that I moved beyond textbook-level knowledge to actual **hands-on coding experience**.

2. To Understand Algorithmic Thinking

Another major objective was to improve my ability to design and implement algorithms. The Tic Tac Toe project required:

- **Win condition detection** algorithms using logical comparisons across rows, columns, and diagonals.
- **Computer move selection** algorithms (randomized but structured to select available cells).
- **Undo operation algorithm**, where the last saved state is retrieved and reapplied.

Through this, I learned how even a simple problem could demand multiple algorithms working together.

3. To Apply DSA in Problem-Solving

A central goal was to highlight how **DSA is not restricted to competitive coding problems** but can be the backbone of even small interactive applications. By implementing this project, I was able to:

- Use data structures to manage game state.
- Apply logical flow control to simulate human-computer interaction.
- Ensure that operations like undo and reset are efficient and error-free.

4. To Enhance UI/UX Creativity with Logic

While DSA provided the backbone of the project, an equally important objective was to **make the project engaging and user-friendly**. To achieve this:

- The project included **color differentiation for X and O**, enhancing clarity for players.
- Fun **"taunt" messages** were integrated to add variety and engagement during gameplay.
- A **confetti effect** was included for celebration when a player won.

This objective emphasized that **algorithms and design** should work hand in hand to create not just functional, but also appealing applications.

5. To Develop Independent Problem-Solving Skills

An important training goal was to **minimize reliance on pre-written code** and instead practice independent logic-building. Challenges such as handling undo restrictions, preventing repeated taunts, and ensuring computer moves were calculated correctly, pushed me to think critically and solve problems step by step.

6. To Prepare for Future Career Demands

In today's industry, strong DSA knowledge is considered essential for interviews and technical assessments. This project served as a **practical revision exercise** for:

- Arrays and stacks.
- Logical implementation of algorithms.
- Time and space complexity analysis of simple operations.

The project not only solidified my DSA base but also prepared me for **upcoming internships and placements**.

7. To Foster Documentation and Reporting Skills

Apart from coding, a key objective of this training was to learn the importance of **technical documentation**. Writing structured reports, maintaining clean code with comments, and preparing presentations formed a crucial part of the training, ensuring that the work undertaken could be **understood, evaluated, and improved upon by others**.

In conclusion, the objectives of this work were multi-dimensional, blending **academic learning (DSA concepts)** with **practical skills (project building)** and **professional skills (documentation, presentation, teamwork)**. By the end of the training, I was able to **achieve these objectives successfully** and gain confidence in applying my classroom learning to real-world scenarios.

1.2 Scope of the Work

The scope of the project defines the **extent, boundaries, and potential impact** of the work undertaken during the summer training. Since the primary objective of the training was to apply **Data Structures and Algorithms (DSA)** in a practical context, the scope was carefully designed to remain **achievable within the time frame of the summer training** while still being **challenging enough to reflect academic and industry relevance**.

1. Academic Scope

The project primarily served as a **practical extension of the Data Structures and Algorithms subject** taught in the curriculum. While classroom learning focuses on theoretical explanations of concepts such as arrays, stacks, queues, linked lists, trees, and graphs, this project allowed for their **application in a real-world scenario**.

- The **array** was implemented to represent the Tic Tac Toe board.
- A **stack-like logic** was applied for maintaining move history to enable the undo feature.
- **Objects and mapping functions** were used to manage states and taunts dynamically.

Thus, the project became a bridge between **theory and practice**, demonstrating how core data structures can serve as the foundation of interactive applications.

2. Practical Scope

From a practical standpoint, the project aimed to create a **fully functional Tic Tac Toe game** with both single-player (vs. computer) and two-player modes.
Key features falling under this scope included:

- **Game mechanics** such as alternating turns, detecting wins, and handling draws.
- **Undo operation** (restricted to computer mode) using move history.
- **Engaging user interface**, enhanced by colors, animations, taunt messages, and confetti celebrations.
- **Error handling and restrictions** (e.g., undo only once, no repeated taunts, preventing overwriting of cells).

The practical scope ensured that the project was not just an academic experiment, but also an **enjoyable, usable application**.

3. Technical Scope

The project used a combination of **front-end technologies** and **algorithmic logic**.

- **Languages used**: HTML, CSS, and JavaScript.
- **DSA integration**: Arrays, stack-like undo mechanism, conditional algorithms for checking wins and moves.
- **Time complexity considerations**: Win-checking runs in constant time since the board size is fixed; move selection runs in linear time relative to empty cells.

This technical scope provided exposure to **software development practices** like modular coding, event handling, DOM manipulation, and maintaining clean project structure.

4. Professional Scope

Beyond academic and technical learning, the project expanded into professional growth areas:

- **Problem-solving and critical thinking**: Finding efficient ways to represent game state and implement undo.
- **User Experience (UX)** awareness: Understanding that algorithms must be complemented by visuals, colors, and engagement to hold user interest.
- **Documentation and presentation**: Preparing this detailed report, coding with comments, and presenting the project formed part of the professional scope, aligning with industry expectations.

5. Future Scope

While the current version of the project fulfills the requirements of the training, its design leaves room for **future enhancements and scalability**. Potential future improvements include:

- **AI-driven computer opponent** using minimax algorithm instead of random moves.
- **Scoreboard system** to track wins, losses, and draws across multiple games.
- **Responsive design** for mobile devices.
- **Multiplayer online mode** using databases and networking concepts.
- **Advanced DSA integration** like trees/graphs for enhanced AI logic.

These possibilities show that the project can evolve into a more sophisticated system, making it suitable for advanced coursework or personal development.

Conclusion of Scope:
The scope of this project was intentionally kept **balanced** — broad enough to cover academic, technical, practical, and professional aspects, yet **focused enough** to be implemented within the duration of summer training. It successfully demonstrates how **DSA concepts can form the backbone of functional applications** and sets the stage for further improvements in the future.

1.3 Importance and Applicability

The importance and applicability of this project can be analyzed across different dimensions — academic, technical, practical, and professional. The purpose of this section is to highlight **why the project matters** and **where the developed skills and concepts can be applied** in real-world scenarios.

1. **Academic Importance**
   - **Practical Understanding of DSA Concepts**
     The Tic Tac Toe game directly demonstrates the use of arrays, stack-based undo logic, condition checking, and mapping — making abstract concepts more tangible.
   - **Bridge Between Theory and Practice**
     Students often study data structures only theoretically; this project shows how the same concepts can power an interactive application.
   - **Foundation for Advanced Topics**
     By implementing simple algorithms here, the student builds confidence to later explore more advanced topics like AI algorithms (e.g., minimax), graph traversal, and optimization problems.

2. **Technical Importance**
   - **Hands-on with Frontend Technologies**
     Using HTML, CSS, and JavaScript to design and implement the game strengthens core web development skills, which are essential in industry.
   - **Algorithmic Thinking**
     The project reinforces step-by-step problem solving, analyzing time complexities, and designing solutions that are not only functional but also efficient.
   - **Event-driven Programming**
     The project uses DOM event listeners to capture user actions, a crucial concept in building interactive applications.

3. **Practical Applicability**
   - **Real-Time Application of DSA**
     While Tic Tac Toe is a small-scale game, the logic used here (board representation, checking winning conditions, handling state transitions) is applicable in larger board games, simulations, and puzzles.

   - **Undo Feature Implementation**
     The undo mechanism mimics stack-based operations and has wide applications in text editors, IDEs, games, and applications where rollback is necessary.

   - **User Engagement Techniques**
     Features like **taunt messages**, **color-coded moves**, and **confetti animations** illustrate how even a simple program can be made engaging for users. Such design awareness is vital for building user-friendly applications.

4. **Professional Applicability**
   - **Industry Relevance of DSA**
     Almost every technical interview in the IT/software sector assesses data structure knowledge. Demonstrating a project based on DSA reflects strong fundamentals.

   - **Team & Client Demonstration**
     A simple, interactive project is easy to showcase to peers, teachers, or clients, making it a good portfolio project for internships and jobs.

   - **Problem Solving Mindset**
     Industry requires not just coding but the ability to design solutions. This project shows how to take a common problem (a game), analyze it, and solve it using DSA principles.

5. **Societal and Educational Applicability**
    - **Educational Tool**
      This project can be used by beginners as a learning aid to understand the application of arrays and condition checking in a fun way.
    - **Entertainment Value**
      Beyond academics, the project has value as a simple web-based game that can be played for entertainment, improving user engagement through gamification.

Conclusion of Importance & Applicability
The project holds importance as it merges academic learning with practical application, showcases technical competency in programming and DSA, and develops skills that are directly applicable in higher studies, industry projects, and professional careers. Its simplicity makes it approachable, while its scope for enhancements makes it a strong foundation for more complex developments.

1.4 Role and Profile

1.9.1 Role of the Trainee

As a student of **B.Tech (CSE) specializing in Cloud Computing** at **Lovely Professional University (LPU)**, my role during the summer training was to **apply theoretical knowledge of Data Structures and Algorithms (DSA)** into practical scenarios. The main focus of my role can be described as follows:

- **Learner and Implementer**
  My primary responsibility was to learn the working of different data structures (arrays, stacks, queues, linked lists, trees, graphs) and apply them in small-scale coding problems as well as a mini-project.

- **Problem Solver**
  Instead of memorizing algorithms, I had to **analyze problems step by step** and decide which data structure was most suitable for solving a given scenario efficiently.

- **Developer**
  To solidify the learning, I developed a **Tic Tac Toe Game using HTML, CSS, and JavaScript** with a strong emphasis on DSA concepts like **arrays (for board representation)**, **stack operations (for undo functionality)**, and **mapping structures (for taunt messages)**.

- **Self-Manager**
  Since the training was self-paced under faculty guidance, I had to manage my own time, break down the learning modules, practice coding problems daily, and ensure that the final project was completed on schedule.

- **Communicator and Presenter**
  Apart from coding, my role also included preparing reports, documenting my work, and being able to present and explain the project during the viva and evaluation process.

1.9.2 Profile of the Training Domain

The training domain selected for the summer training was **Data Structures and Algorithms (DSA)**.
The profile of this domain is as follows:

- **Fundamental Area of Computer Science**
  DSA is the backbone of computer science and software engineering. It deals with **organizing data (data structures)** and **performing operations efficiently (algorithms)**.

- **Core Concepts Covered**
  - **Arrays**: For storing and manipulating sequential data.
  - **Stacks & Queues**: For order-based operations and undo/redo logic.
  - **Linked Lists**: For dynamic memory management and node-based data handling.
  - **Trees**: For hierarchical representation and searching.
  - **Graphs**: For relationships and networking problems.
  - **Sorting & Searching Algorithms**: For optimization and efficiency in data handling.

- **Industry Importance**
  - Almost every software system relies on efficient data handling — from databases to operating systems to cloud platforms.
  - DSA concepts form the foundation of coding interviews, competitive programming, and real-world applications.

- **Profile in the Context of the Project**

  In the context of my project (Tic Tac Toe), the training profile focused on:
  - Representing the game board using arrays.
  - Checking winning conditions using iteration and logical conditions.
  - Implementing **Undo feature** using stack-like behavior.

- Enhancing gameplay experience with **object-based mapping** for taunt memory.

Conclusion

Thus, my **role** was that of a learner, problem-solver, and developer, while the **profile of the training domain** emphasized the importance of DSA as a fundamental skill set for computer science students. This training experience not only enhanced my technical knowledge but also improved my analytical thinking, self-learning ability, and project development skills.

Chapter 2: INTRODUCTION OF THE COMPANY/WORK

2.1 Company's Vision and Mission

2.1.1 Vision of Lovely Professional University (LPU)

Lovely Professional University envisions itself as a **leading global institution**, recognized for its academic excellence, research-driven innovation, and commitment to preparing students for **21st-century challenges**.

The vision emphasizes:

- **Transforming Education**: To redefine higher education through innovative teaching methodologies, industry-integrated curriculum, and digital learning platforms.
- **Producing Global Leaders**: To nurture professionals equipped with not only technical knowledge but also leadership, ethics, and values.
- **Research and Innovation**: To be a hub of advanced research in science, technology, and interdisciplinary domains, thereby contributing to national and international development.
- **Inclusivity and Diversity**: To provide a multicultural environment where students from all backgrounds can thrive and learn.
- **Sustainability and Social Responsibility**: To create socially responsible graduates who work towards environmental sustainability and community development.

In essence, LPU's vision is to **become a globally respected academic institution that creates professionals capable of driving innovation, entrepreneurship, and societal transformation**.

2.1.2 Mission of Lovely Professional University (LPU)

The mission of LPU reflects its **core values and operational goals** aimed at fulfilling the university's vision. The mission can be summarized as follows:

1. **Deliver Quality Education**
   To impart **world-class, application-oriented education** in diverse fields of study while focusing on both technical and life skills development.

2. **Foster Research and Innovation**
   To encourage a culture of **research, critical thinking, and problem-solving**, ensuring that students can innovate and contribute to technological advancements.
3. **Strengthen Industry-Academia Linkages**
   To create strong connections with industries, enabling students to gain **practical exposure, internships, and placements** that align with global job markets.
4. **Encourage Entrepreneurship**
   To develop an ecosystem that supports **startups, incubators, and entrepreneurial mindsets**, helping students convert ideas into real-world solutions.
5. **Promote Global Outlook**
   To provide **international collaborations, exchange programs, and diverse learning opportunities**, ensuring students are globally competitive.
6. **Value-Based Education**
   To instill **ethics, discipline, and social responsibility**, making graduates not only successful professionals but also responsible citizens.

Conclusion
The **vision and mission of Lovely Professional University** align strongly with the objectives of summer training programs like this one. By focusing on **Data Structures and Algorithms (DSA)**, this project directly reflects LPU's mission of promoting **skill development, research, problem-solving ability, and innovation** in students.

2.2 Origin and Growth of Company

2.2.1 Origin of Lovely Professional University (LPU)

Lovely Professional University (LPU) is a premier institution of higher learning established under the **Lovely Professional University Act, 2005** by the State Legislature of Punjab. The foundation of LPU was laid by the **Lovely Group**, which initially started as a small educational venture in 1961 with a single school. Over time, this vision of providing quality education expanded into a full-fledged university.

The university officially began operations in **2006**, and since then, it has emerged as one of the **largest private universities in India**, both in terms of student strength and campus infrastructure. Located in **Phagwara, Punjab**, LPU was envisioned to provide **world-class education, industry exposure, and global opportunities** to Indian students at affordable costs.

2.2.2 Growth and Development of LPU

Since its establishment, LPU has witnessed **tremendous growth and recognition** both nationally and internationally. The university's growth journey can be categorized in the following phases:

1. Academic Expansion

- Started with a limited number of undergraduate programs in 2006.
- Today, it offers **200+ programs** across disciplines including **Engineering, Computer Science, Management, Law, Pharmacy, Agriculture, Design, and Humanities**.
- Known for **industry-aligned curriculum** and specialization programs (like Cloud Computing, AI, Cybersecurity), ensuring students meet modern industry demands.

2. Student Strength

- In its initial years, the university admitted only a few thousand students.
- Currently, LPU hosts more than **35,000 on-campus students** and has **a vast alumni network across 50+ countries**.
- It is recognized as one of the largest residential universities in India with students from **70+ countries**, giving it a multicultural environment.

3. Infrastructure Growth

- The LPU campus spans over **600 acres**, with **state-of-the-art academic blocks, libraries, research centers, and digital learning hubs**.
- Advanced **labs for engineering and computer science**, including **AI labs, cloud computing labs, and cybersecurity labs**, have been set up to keep pace with emerging technologies.

4. Research and Innovation

- LPU has significantly increased its research output, with **patents filed, publications in reputed journals, and funded projects** in diverse fields.
- It encourages innovation through **incubation centers** and partnerships with industries and startups.

5. National and International Recognition

- Ranked among the **top private universities in India** by NIRF (National Institutional Ranking Framework).

- Partnered with **Harvard Business School Online, Google Cloud, Microsoft, Cisco, and Amazon Web Services** for academic collaborations.
- Recognized globally with **tie-ups with 200+ international universities** for student exchange and research collaborations.

2.2.3 Contribution to Students and Society

LPU's growth is not limited to infrastructure and academics alone. Its emphasis on **skill development, innovation, social responsibility, and global exposure** has created thousands of successful graduates working in **Fortune 500 companies, government organizations, and entrepreneurial ventures**.

The university has also contributed to society by:

- Offering **scholarships worth hundreds of crores** to deserving and economically weak students.
- Running initiatives for **community development, sustainability, and rural outreach**.

The journey of Lovely Professional University from a small initiative by the Lovely Group to a globally recognized institution has been remarkable. Its consistent **growth in academics, research, student strength, and industry partnerships** reflects its commitment to becoming a **world-class university**.

Chapter 3: TRAINING OVERVIEW

3.1 Overview

The summer training undertaken during this semester was entirely focused on the subject **Data Structures and Algorithms (DSA)**. The aim of this training was to strengthen problem-solving skills, improve logical reasoning, and gain practical exposure by implementing concepts through a real-world project. Since DSA is considered the backbone of computer science, the training provided both **theoretical foundation** and **hands-on practice** with key data structures like arrays, linked lists, stacks, queues, trees, and graphs.

The training also emphasized how these abstract concepts can be applied in designing efficient solutions for problems such as games, search engines, scheduling, and simulations. My project, **"Tic-Tac-Toe Game,"** served as a practical demonstration of these concepts in action.

3.1 Literature Survey / Related Work

Games like Tic-Tac-Toe have been widely studied in the field of computer science and artificial intelligence because of their simplicity and clear problem structure. They serve as an excellent introduction to the application of Data Structures and Algorithms in gaming logic.

1. **Tic-Tac-Toe in Artificial Intelligence Education:**
   Numerous academic studies and tutorials have used Tic-Tac-Toe as the first step to explain the Minimax algorithm, a classic recursive approach for decision-making in zero-sum games. It demonstrates how trees, recursion, and backtracking work in practice.
2. **Use of Stack for Undo Operations:**
   The concept of using stacks for undo/redo functionality is a standard practice in software systems, such as text editors and design tools. Implementing this in a game project not only strengthens understanding of the stack data structure but also shows practical real-world relevance.
3. **Related Small Games and DSA Concepts:**
   Similar small-scale games, such as Connect Four or Sudoku solvers, also employ arrays, trees, and backtracking. Tic-Tac-Toe stands out as it can be implemented using minimal resources while still demonstrating the fundamentals of DSA like arrays, maps, and stacks.
4. **Tic-Tac-Toe as a Problem in Game Theory:**
   In mathematical and AI literature, Tic-Tac-Toe is a solved game—meaning the outcome

can always be predicted with optimal play. This makes it a good example to teach optimal strategies, state-space representation, and game trees.

5. **Unique Contribution of This Project:**
   While most academic or tutorial-based implementations focus only on the Minimax approach, this project emphasizes creativity and user engagement. The addition of stack-based undo, randomized computer moves, UI enhancements, and interactive taunts makes the project both educational and fun, bridging the gap between theory and practical application.

3.2 Tools and Technologies Used

During the course of training, the following tools and technologies were used:

1. **Programming Languages:**
   - JavaScript: Used as the primary language for implementing game logic and DSA concepts.
   - HTML & CSS: For creating the front-end interface of the project.
2. **Platforms & Editors:**
   - Visual Studio Code (VS Code): Used as the development environment.
   - GitHub & Netlify: For version control and project deployment.
3. **Supporting Tools:**
   - Confetti.js library: Added for fun animation effects during winning events.
   - Browser Console / Debugger: For testing and debugging the implemented code.

3.3 Areas Covered During Training

The training primarily covered the following core data structures and algorithms:

1. **Arrays**
   - Used to represent the game board in Tic-Tac-Toe as a 1-D array of size 9.
   - Time Complexity for accessing/modifying a cell: O(1).
   - Practical learning: How arrays provide direct indexing and fast access.
2. **Linked List (theoretical coverage with examples)**
   - Understood as a linear collection of nodes where each node points to the next.

    - Compared with arrays in terms of insertion/deletion efficiency.
    - Though not directly applied in project, practiced through exercises during training.

3. **Stack**
    - Learned as a LIFO (Last-In-First-Out) structure.
    - Applied in the Undo feature of the project: previous game states are stored, and the most recent state can be retrieved.
    - Operations Time Complexity: Push/Pop = O(1).
4. **Queue**
    - Understood as FIFO (First-In-First-Out) structure.
    - Applied conceptually in scenarios like player turn handling and event scheduling.
5. **Trees & Binary Search Trees (BST)**
    - Understood hierarchical representation of data.
    - Practiced examples: searching, inserting, and traversing nodes.
    - Though not directly used in Tic-Tac-Toe, the concept of "searching optimal moves" relates to decision trees (like in Minimax algorithm).
6. **Graphs**
    - Learned about nodes and edges representation.
    - Understood real-life applications like shortest paths, networks, and relationships.
    - Tic-Tac-Toe board can also be theoretically modeled as a graph of 9 nodes connected by winning edges.

3.4 Daily / Weekly Work Summary

The training was structured over several weeks, with consistent progress from fundamentals to project implementation:

- **Week 1:**
  Introduction to Data Structures, importance of DSA in problem solving, revision of Arrays with small coding exercises.
- **Week 2:**
  Learning Stacks and Queues with coding problems; practice of push / pop / enqueue / dequeue operations.
- **Week 3:**
  Linked List (singly, doubly), Trees (binary and BST), Graphs (BFS, DFS). Solved multiple basic problems from each topic.

- **Week 4:**
  Applied knowledge into project planning. Designed Tic-Tac-Toe board using arrays, studied how stack can be used for Undo functionality, and learned how decision trees could make a smarter computer player.
- **Week 5:**
  Project development phase:
  - Created user interface using HTML/CSS.
  - Implemented game logic in JavaScript.
  - Added features like Undo, vs Computer mode, Taunts, and Confetti animation.
- **Week 6:**
  Testing, debugging, and final deployment of the project on **Netlify**. Prepared documentation and report writing for submission.

3.5 Applications of Data Structures and Algorithms in Real Life

Data Structures and Algorithms (DSA) form the backbone of computer science and modern technology. They provide systematic methods to organize, store, and retrieve data efficiently while enabling problem-solving strategies that ensure optimal performance. During my summer training, I not only applied DSA concepts in my Tic-Tac-Toe project but also realized their broader applications in real-world systems. Some of the most significant applications are discussed below:

1. Arrays and Strings

Arrays and strings are the simplest data structures but are fundamental in all applications of computing.

- **Use Cases:**
  - Image processing (storing pixel values in arrays).
  - Multimedia storage (audio, video streams are stored as arrays of bytes).
  - Search operations in spreadsheets, databases, and text editors.
  - Game development, where the board (such as in Tic-Tac-Toe) is represented as an array.

2. Stacks

A stack follows the **LIFO (Last In First Out)** principle and is widely used where reversal or undo functionality is needed.

- **Use Cases:**
    - Undo/Redo features in editors (Microsoft Word, Google Docs).
    - Browser history (back and forward navigation).
    - Expression evaluation (converting infix to postfix).
    - Recursion in programming is implemented internally using stack memory.
    - In my project, stack logic was used to store board states for the **Undo feature** against the computer.

3. Queues

Queues follow the **FIFO (First In First Out)** principle, ensuring fairness in order processing.

- **Use Cases:**
    - CPU scheduling in operating systems (Round Robin scheduling uses queues).
    - Print job management in printers.
    - Call center ticketing systems.
    - Messaging services and real-time data streaming (e.g., Kafka).

4. Linked Lists

Linked lists allow dynamic memory allocation and efficient insertion/deletion.

- **Use Cases:**
    - Music playlists and image slideshows (where next/previous navigation is required).
    - Dynamic memory allocation in operating systems.
    - Implementation of stacks, queues, and adjacency lists in graphs.
    - Blockchain technology uses linked structures for chaining blocks securely.

5. Trees

Tree structures enable hierarchical data organization.

- **Use Cases:**
    - File system organization (folders and subfolders).
    - XML/HTML parsing in browsers.
    - Databases use **B-trees** and **B+ trees** for indexing.

  - Search engines use **tries** for fast autocomplete suggestions.
  - AI in games (like Chess or advanced Tic-Tac-Toe) uses **game trees** for decision-making.

6. Graphs

Graphs are powerful for modeling relationships and networks.

- **Use Cases:**
  - Social media platforms (Facebook friend graph, LinkedIn connection graph).
  - Google Maps for shortest path calculations (Dijkstra's algorithm).
  - Network routing protocols in the internet.
  - Recommendation systems (Netflix suggesting movies based on user preferences).

7. Hashing

Hashing is used to map keys to values with high efficiency.

- **Use Cases:**
  - Database indexing.
  - Password storage and verification.
  - Caching mechanisms in web applications.
  - Compiler symbol tables.

Conclusion of Applications

From simple arrays that help represent a game board to advanced graph algorithms that power Google Maps, DSA is everywhere in real life. This shows that even a small project like Tic-Tac-Toe is not just a game but a practical demonstration of how these concepts form the foundation of advanced computer applications. Learning DSA through this training is therefore not only an academic requirement but also an essential step towards building strong problem-solving skills for real-world applications.

Conclusion of Chapter 3

The training provided a comprehensive understanding of **DSA concepts** and their applications in real-world scenarios. By the end of the training, I not only revised the fundamentals of arrays, stacks, queues, linked lists, trees, and graphs but also applied them practically through my project. This helped bridge the gap between **theoretical learning** and **practical problem solving**.

Chapter 4: PROJECT DETAILS

4.1 Title of the Project

"Tic-Tac-Toe with Undo and Computer Play (A DSA-Based Approach)"

Project Link:- https://tic-tac-toe-dsa.netlify.app/

4.2 Problem Definition

Games are not only a source of entertainment but also a practical way to apply programming concepts and logical thinking. **Tic-Tac-Toe** is one of the simplest yet most popular strategy games played worldwide.

However, in a basic Tic-Tac-Toe implementation:

- There is **no undo option**, making mistakes permanent.
- Playing against another player is possible, but the challenge is limited without a computer opponent.
- The game often lacks fun elements like animations or engaging user feedback.

The problem, therefore, is:

- To design and implement a Tic-Tac-Toe game that not only allows **two players to compete** but also provides the option to **play against a computer opponent**.
- To integrate **DSA concepts**, such as **Stack** (for Undo) and **Array** (for the board representation).
- To improve the **user experience (UX)** by including features such as **visual highlights, taunt messages, and celebration effects (confetti)**.

This project aims to combine **DSA-driven logic** with **fun game design**, showing the importance of abstract concepts like arrays and stacks in solving real-world interactive problems.

4.3 Scope and Objectives

4.3.1 Scope

- The project demonstrates the **application of core DSA concepts** in a simple but engaging way.
- Provides an **interactive platform** where a user can play in two modes:
    1. Player vs Player
    2. Player vs Computer
- Adds **Undo functionality** to give more control to the player.
- Incorporates **UI/UX design** with animations, colors, and messages to make the game more appealing.
- Can be expanded in future versions to:
    - Include difficulty levels (Easy, Medium, Hard) using **Tree-based Minimax Algorithm**.
    - Maintain a scoreboard using **file/database storage**.
    - Add multiplayer online play.

4.3.2 Objectives

1. To represent the game board using an **Array**.
2. To implement **Undo functionality** using a **Stack**.
3. To allow the game to be played in both **multiplayer** and **computer mode**.
4. To enhance user engagement using **taunt messages, winning highlights, and confetti effects**.
5. To apply **modular programming** principles by dividing the game into clear modules (UI, Logic, Undo, etc.).
6. To strengthen practical knowledge of **DSA and Algorithms** through real implementation.

4.4 System Requirements

4.4.1 Hardware Requirements

- Processor: Intel i3 or higher
- RAM: Minimum 2 GB
- Storage: 100 MB free space

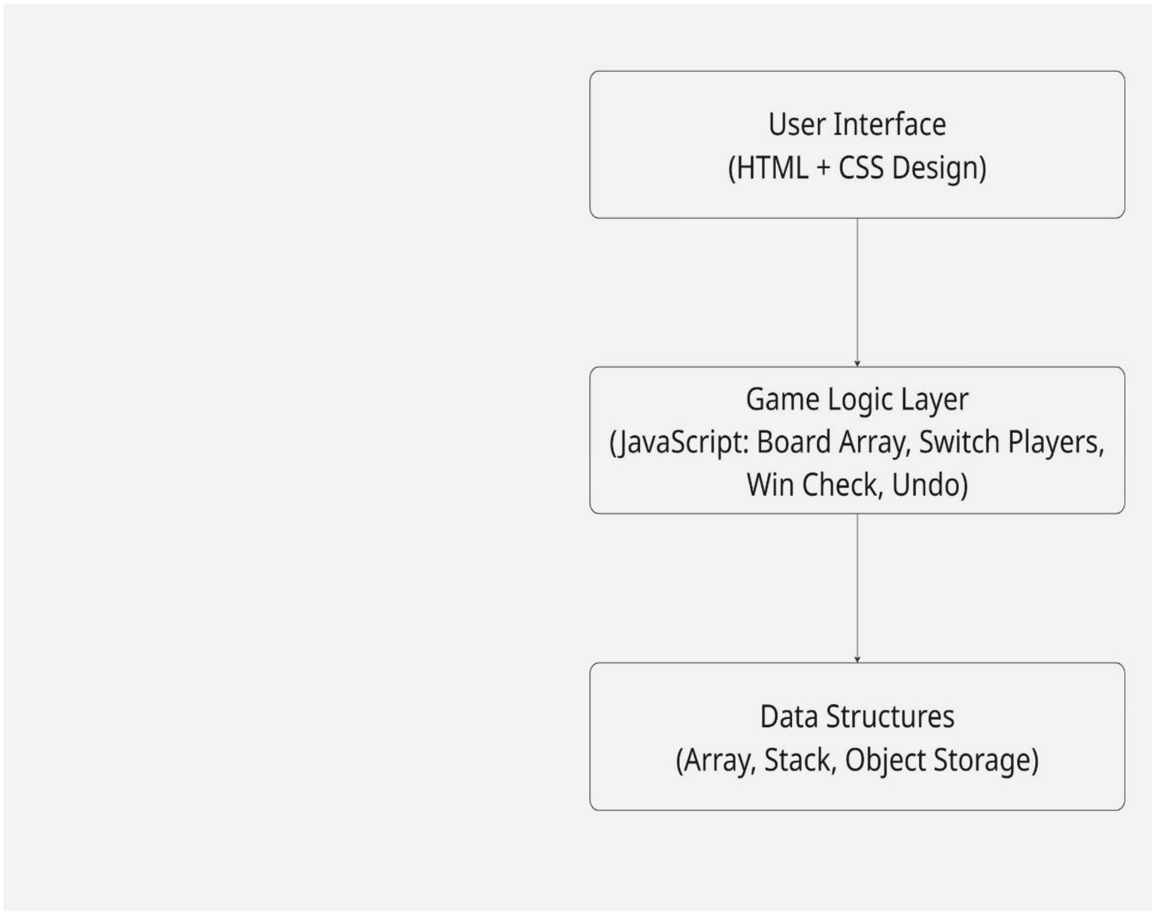
- Display: Standard 13” or above monitor

4.4.2 Software Requirements

- Operating System: Windows / Linux / macOS
- Code Editor: Visual Studio Code (or any text editor)
- Browser: Chrome / Edge / Firefox
- Deployment Tools: GitHub & Netlify (optional for hosting)

4.5 Architecture Diagram

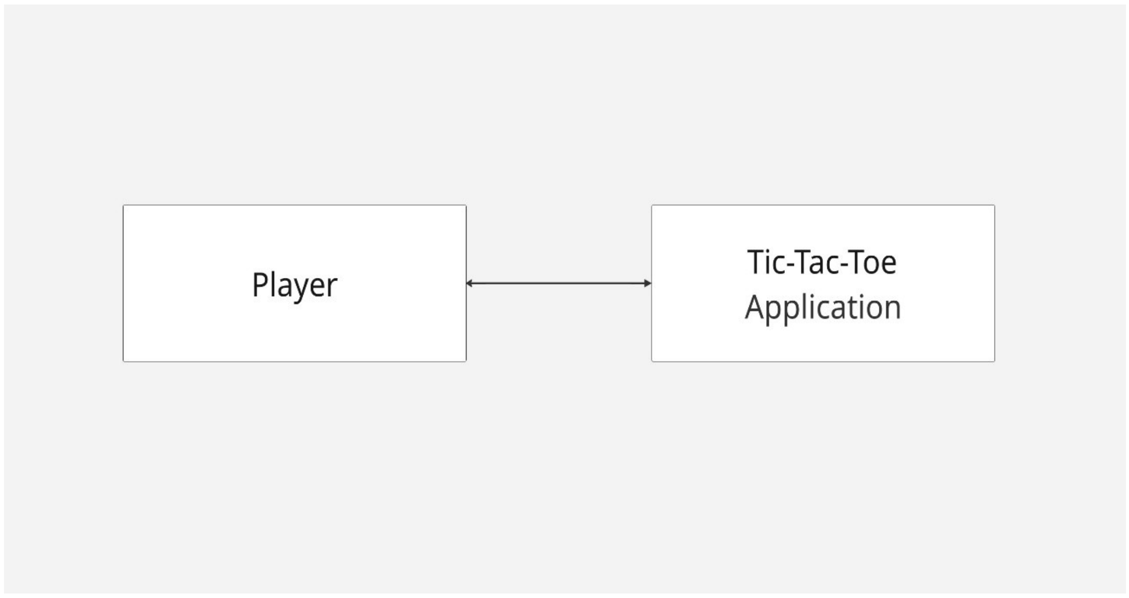
The system is designed in **three layers**:


User Interface
(HTML + CSS Design)
Game Logic Layer
(JavaScript: Board Array, Switch Players, Win Check, Undo)
Data Structures
(Array, Stack, Object Storage)

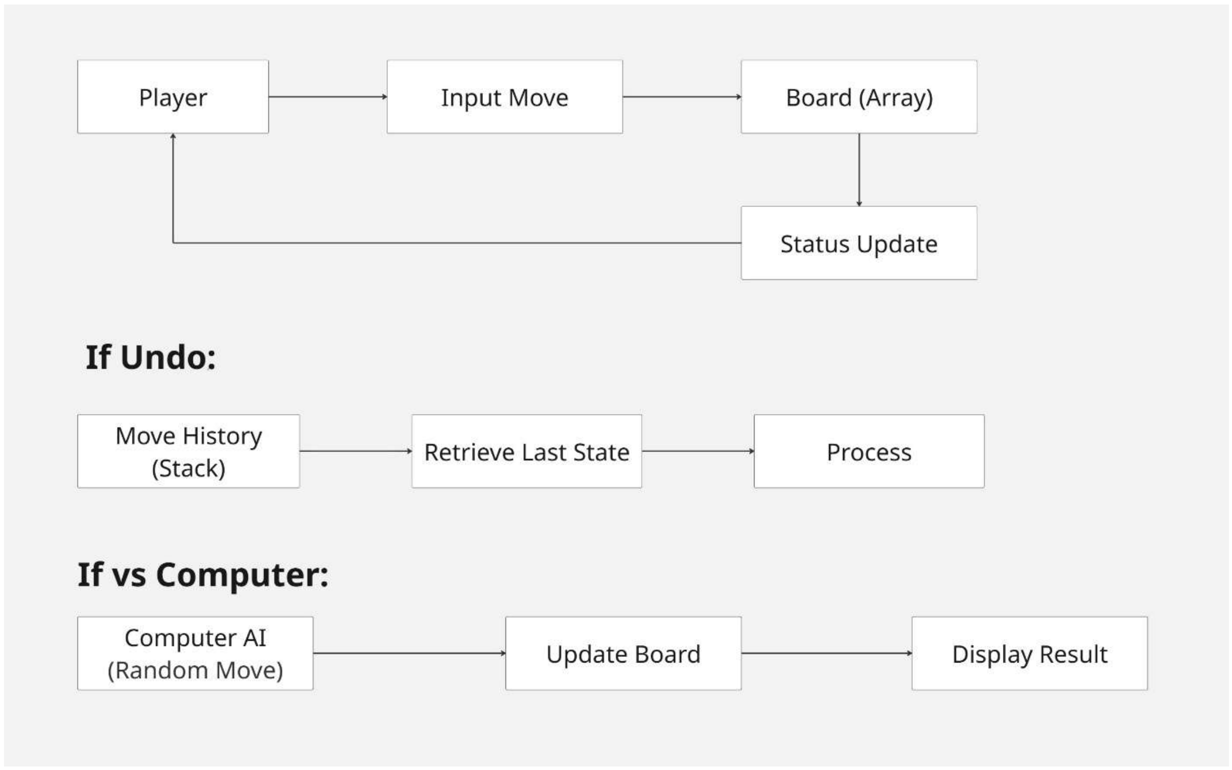
- **User Interface (UI Layer):** Handles how the game looks and interacts with the user.
- **Game Logic Layer:** Implements rules (turns, checking winner, draw).
- **DSA Layer:** Ensures efficient storage and retrieval using Array & Stack.

4.6 Data Flow Diagram (DFD)

4.6.1 Level 0 (Context Diagram):



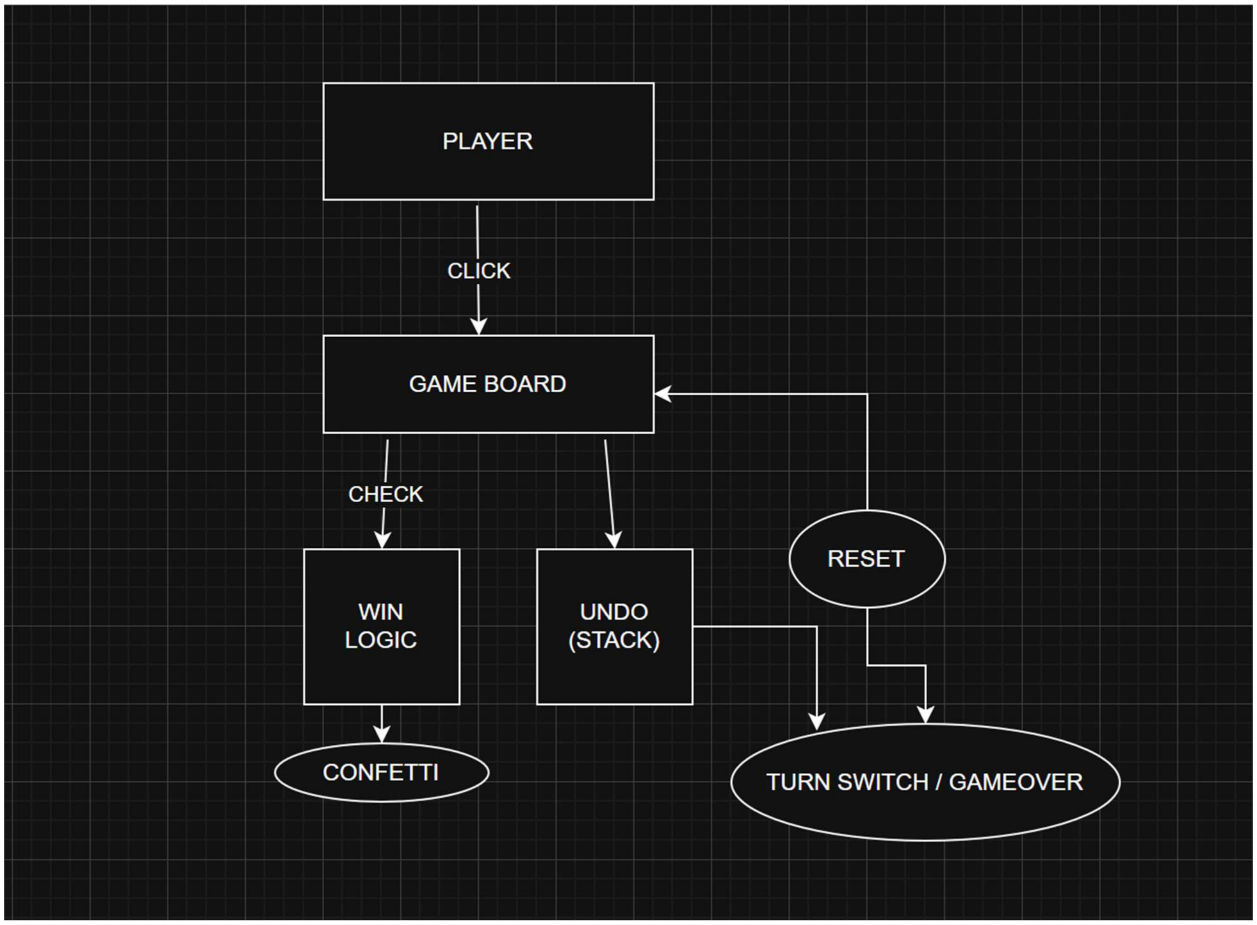
4.6.2 Level 1 (Detailed Flow):



4.7 UML Use Case

- Actors:
  - Player X
  - Player O / Computer
- Use Cases:
  - Start Game
  - Make Move
  - Undo Move (only once per game in vs Computer mode)
  - Check Winner
  - Display Result / Messages
  - Reset Game

4.8 Flowchart



The project successfully bridges **theoretical DSA concepts** and **practical application**. It shows how arrays and stacks can be applied in real-life scenarios like gaming. The design is modular, user-friendly, and demonstrates creativity along with technical understanding.

Chapter 5: IMPLEMENTATION

5.1 Tools Used

1. Programming Languages & Frameworks
   - HTML5: Used for creating the game structure and interface.
   - CSS3: Applied for styling, layout design, and providing the "dashboard look" with neon colors.
   - JavaScript (ES6): Implemented the main game logic, undo feature, taunts, confetti effect, and computer moves.
2. Software Tools
   - Visual Studio Code: IDE used for development.
   - GitHub: Version control and project hosting.
   - Netlify: For deployment and public sharing of the game.
3. Libraries
   - Confetti.js: For celebration effect when a player wins.

5.2 Methodology

The methodology followed in the project was **modular and incremental development**:

1. **Phase 1: Board Setup**
   - Represented the board using an **array of size 9**.
   - Displayed cells on the browser using HTML & CSS.
2. **Phase 2: Player vs Player Mode**
   - Implemented alternating turns between Player X and Player O.
   - Added win condition checks using predefined winning patterns.
3. **Phase 3: Undo Feature (DSA - Stack)**
   - Used a **stack** to save board states.
   - Allowed undo (limited to one move in computer mode).
4. **Phase 4: Computer Mode**
   - Implemented random move selection for the computer.

   - Added delay to simulate “thinking time.”
5. **Phase 5: UI Enhancements**
   - Added different colors for X and O.
   - Highlighted winning cells.
   - Added fun taunt messages and confetti celebrations.
6. **Phase 6: Testing & Deployment**
   - Tested for different use cases (win, draw, undo).
   - Deployed on **Netlify** for public access.

5.3 Modules of the Project

1. **Game Board Module**
   - HTML <div> elements for 9 cells.
   - Array [0–8] to store moves.
2. **Game Logic Module**
   - Alternating turns (X and O).
   - Win/draw detection.
3. **Undo Module (Stack Implementation)**
   - Stores previous state of the board.
   - Allows retrieval when undo is triggered.
4. **Computer Play Module**
   - Identifies empty cells.
   - Selects a random move.
5. **User Interface Module**
   - Status updates with taunt messages.
   - Confetti animation.
   - Highlighting winning cells.

5.4 Code Snippets

Some important parts of the code are included in the report for demonstration of DSA usage:

1. Board Representation using Array

```
let board = ["", "", "", "", "", "", "", "", ""];
```

2. Undo using Stack (Move History)

```
let moveHistory = [];
if (vsComputer && currentPlayer === "X" && !undoUsed) {
    moveHistory.push([...board]);
}
```

3. Undo Logic

```
const lastSavedState = moveHistory.pop();
board = [...lastSavedState];
updateBoardUI();
```

4. Win Condition Check

```
const winConditions = [
  [0, 1, 2], [3, 4, 5], [6, 7, 8],
  [0, 3, 6], [1, 4, 7], [2, 5, 8],
  [0, 4, 8], [2, 4, 6]
];
for (let condition of winConditions) {
  const [a, b, c] = condition;
  if (board[a] && board[a] === board[b] && board[a] === board[c]) {
      // Winner Found
  }
}
```

5. Computer Random Move

```
let emptyCells = board.map((val, i) => val === "" ? i : null).filter(i => i !== null);
const randomIndex = emptyCells[Math.floor(Math.random() * emptyCells.length)];
makeMove(randomIndex, "O");
```

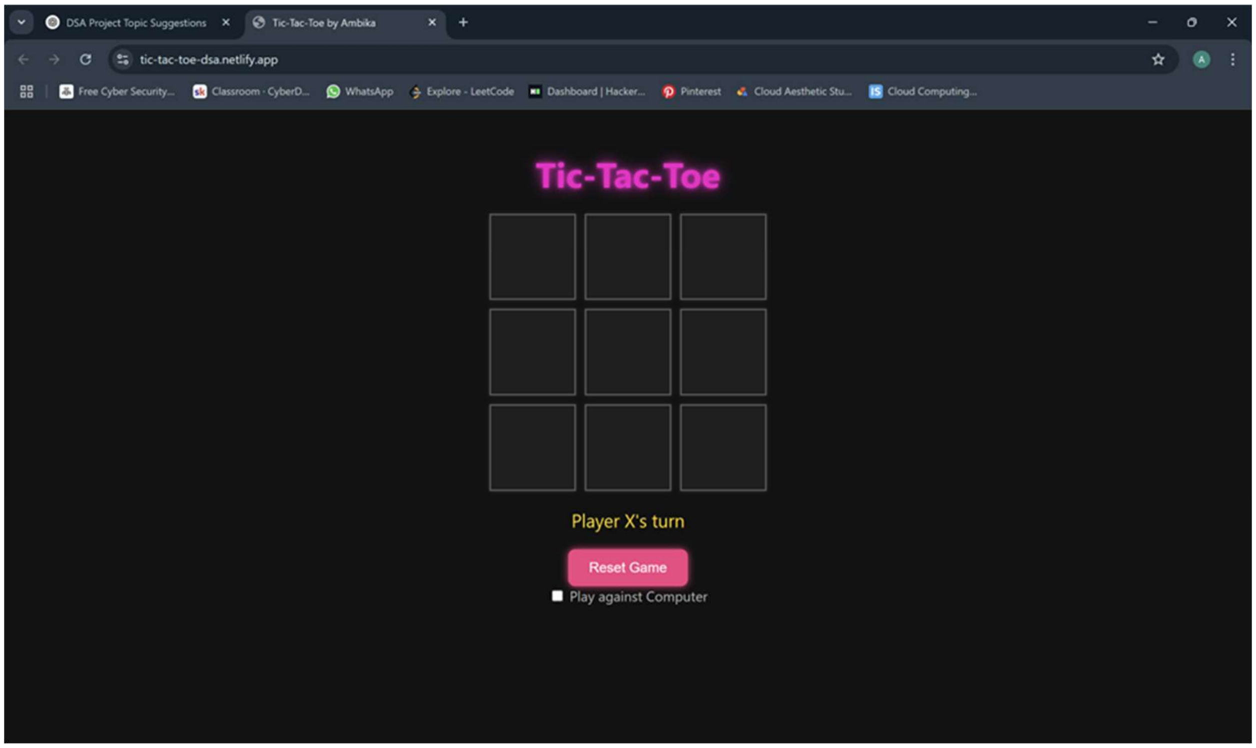
Chapter 6: RESULTS and DISCUSSION

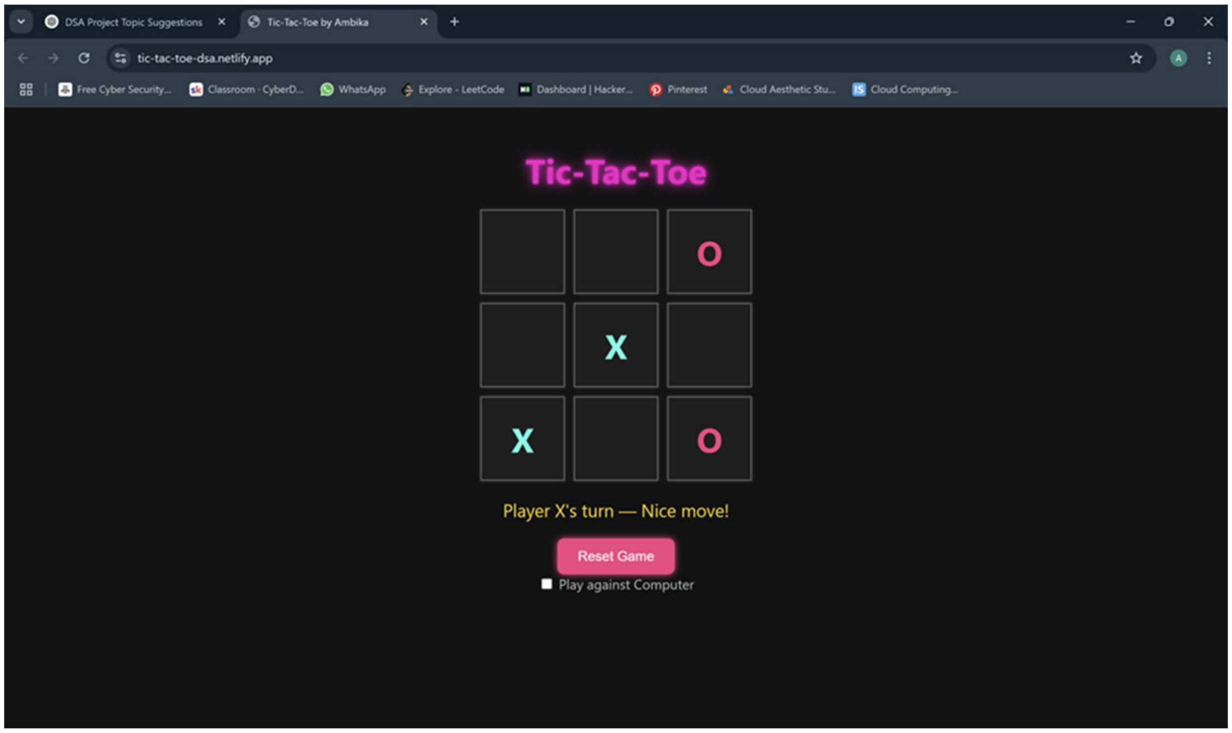
6.1 Output / Report

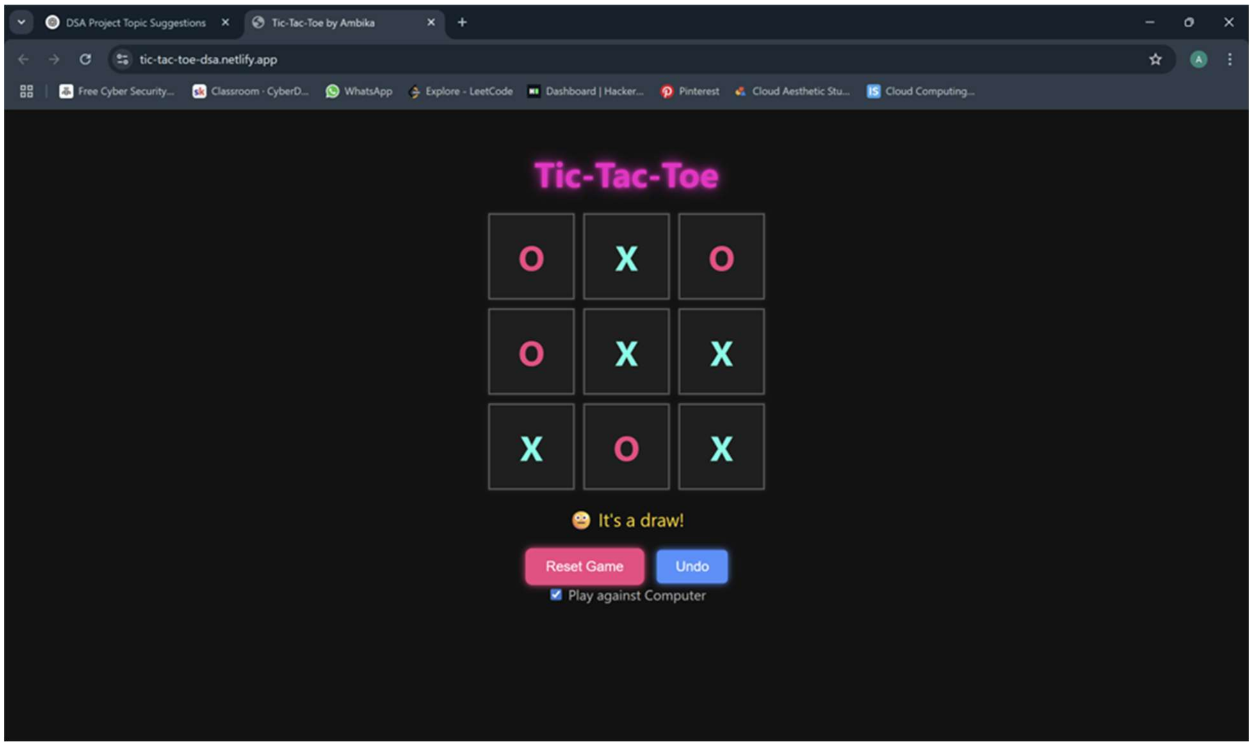
The developed project "Tic-Tac-Toe with Undo Feature and Computer Mode" produced the expected results successfully. The following key outputs were observed:

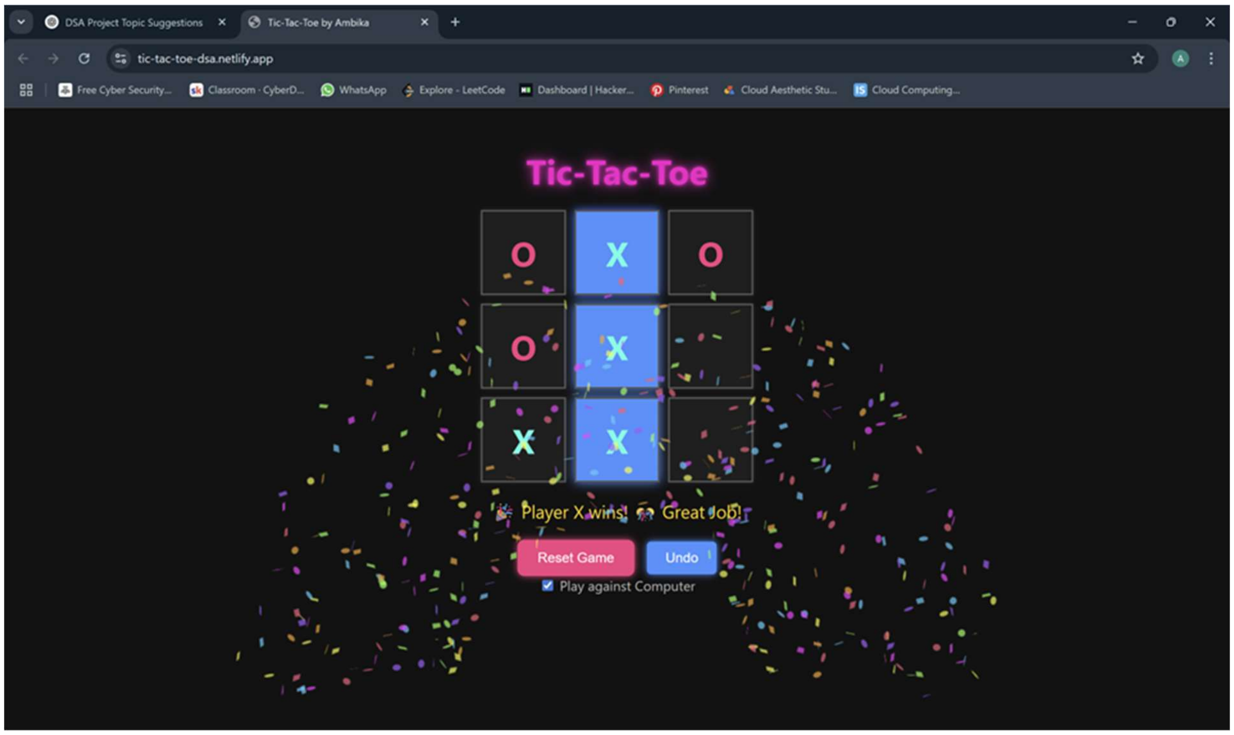
1. Player vs Player Mode
   - Two human players can alternately play the game.
   - The game correctly detects win, loss, or draw.
   - Winning cells are highlighted with a different color.
2. Player vs Computer Mode
   - The user plays as Player X, while the computer plays as Player O.
   - The computer move is generated randomly with a slight delay to simulate thinking time.
   - Undo feature allows the player to cancel one move per game, providing a strategic advantage.
3. Undo Feature (Stack Implementation)
   - Undo can only be used once in vs Computer mode.
   - Board state is restored correctly after undo.
   - Alert message is shown if the player tries to use undo more than once.
4. Taunts and Celebrations
   - After each move, a fun taunt message is displayed (e.g., "Nice move!", "Boom!", "Tension rising…").
   - On winning, confetti animation is triggered, making the game more interactive.
5. Draw Detection
   - If all cells are filled and no winning condition is met, the system correctly declares a draw.

6.2 Screenshots






DSA Project Topic Suggestions
Tic-Tac-Toe by Ambika
tic-tac-toe-dsa.netlify.app
Free Cyber Security...
Classroom · CyberD...
WhatsApp
Explore - LeetCode
Dashboard | Hacker...
Pinterest
Cloud Aesthetic Stu...
Cloud Computing...
Tic-Tac-Toe
O
X
O
O
X
X
X
O
X
It's a draw!
Reset Game
Undo
Play against Computer


DSA Project Topic Suggestions
Tic-Tac-Toe by Ambika
tic-tac-toe-dsa.netlify.app
Tic-Tac-Toe
O
X
O
O
X
X
X
Player X wins! Great Job!
Reset Game
Undo
Play against Computer

6.3 Challenges Faced

During the implementation of this project, several challenges were encountered:

1. **Integrating Undo Feature with Computer Mode**
    - Initially, the undo operation removed only the computer's move instead of the player's.
    - Solved by introducing a stack to store board states before player moves.
2. **Avoiding Repetition in Taunts**
    - The same taunt message was being repeated frequently.
    - Solved by maintaining a taunt memory object to avoid repetition.
3. **Highlighting Winning Cells**
    - At first, it was not visually clear which player won.
    - Solved by applying CSS class "winning-cell" to highlight the winning combination.
4. **UI/UX Design**
    - The initial design looked too simple and not like a game.
    - Solved by applying dark dashboard theme, neon colors for X and O, and interactive hover effects.
5. **Deployment Issues**
    - During hosting, relative file paths for CSS/JS caused issues.
    - Solved by ensuring correct project structure before deploying on Netlify.

6.4 Learnings

From this project, several important technical and non-technical learnings were achieved:

1. **Data Structures & Algorithms (DSA)**
    - Learned practical implementation of **stack** for undo operations.
    - Used **array** effectively for representing the game board.
    - Understood how to apply win conditions using **pattern-matching logic**.
2. **Problem-Solving Skills**
    - Gained experience in debugging logical errors.
    - Learned how to break down a big problem into smaller modules.
3. **UI/UX Design Concepts**

- Importance of color contrast and layout for user engagement.
- Enhanced skills in creating **interactive dashboards**.

4. **Web Development Skills**
    - Improved knowledge of **JavaScript event handling**.
    - Practical exposure to **deployment on Netlify** and version control using **GitHub**.
5. **Soft Skills**
    - Time management in developing within deadlines.
    - Ability to explain technical logic in simple words (helpful for viva and presentations).

CONCLUSION and FUTURE SCOPE

Conclusion

The project "Tic-Tac-Toe with Undo Feature and Computer Mode" was successfully designed and implemented as part of the Summer Training program in the subject of Data Structures and Algorithms (DSA).

Through this project, a simple childhood game was reimagined into an interactive, modern, and engaging platform by applying fundamental programming and DSA concepts. Arrays were used to represent the board, stacks were employed to implement the undo functionality, and conditional logic was applied for checking win conditions. The project not only served as a strong application of DSA principles but also provided practical exposure to web development using HTML, CSS, and JavaScript.

Key achievements of this project include:

- Player vs Player Mode for interactive play between two users.
- Player vs Computer Mode with automated moves.
- Undo functionality for strategic gameplay (implemented using stack).
- Winning highlights and confetti animations to improve user engagement.
- Randomized taunt messages to make the game fun and less monotonous.

The completion of this project also enhanced my confidence in bridging the gap between theoretical learning of algorithms and their real-world implementation. It has also developed my skills in UI/UX design, coding best practices, debugging, and deployment on online platforms like Netlify.

Future Scope

While the current implementation of the Tic-Tac-Toe game with stack-based undo, randomized computer moves, taunts, and celebratory animations provides a complete and enjoyable user experience, there remains significant scope for future improvements and expansion. Some possible enhancements include:

1. **Difficulty Levels using AI (Minimax Algorithm):**
   The current computer player operates on random move selection. By implementing algorithms such as Minimax with Alpha-Beta Pruning, the computer can simulate an intelligent opponent that plays optimally. This will allow users to select difficulty levels (Easy, Medium, Hard) and make the game more challenging.

2. **Online Multiplayer Mode:**
   Extending the project to support real-time online gameplay using WebSockets, Firebase, or similar technologies can allow users to play against friends remotely. This would increase engagement and make the project more applicable in real-world scenarios.
3. **Persistent Scoreboard and Player Profiles:**
   At present, the game resets scores on refresh. Future versions can integrate databases or local storage to save match history, player profiles, and scores. This would simulate a tournament-style system.
4. **Graphical Enhancements and Sound Effects:**
   The game can be enhanced with themes, background music, and sound effects for moves and wins. Adding animations or GIFs can make the user experience more interactive and fun.
5. **Mobile Application Version:**
   Developing the same game as a mobile app using React Native or Flutter would expand its accessibility, allowing users to play seamlessly on smartphones.
6. **Extension to Other Games:**
   The concepts used in this project—stack for undo, randomization, and win condition checks—can be extended to slightly more complex board games such as Connect Four, Sudoku validation, or Checkers. This would further highlight the strength of Data Structures and Algorithms in real-world gaming logic.

Thus, the project provides a strong foundation that can evolve into an advanced AI-driven, multi-platform, and feature-rich application.

Final Note

This project demonstrated that even a simple game like Tic-Tac-Toe can serve as a powerful platform to learn and apply core concepts of Data Structures and Algorithms. With further improvements and scaling, it has the potential to become not just a learning exercise, but also a fully engaging and competitive game for broader audiences.

LOVELY PROFESSIONAL UNIVERSITY

REFERENCES

1. Mozilla Developer Network, “JavaScript Guide and Reference,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

2. GeeksforGeeks, “Data Structures and Algorithms,” [Online]. Available: https://www.geeksforgeeks.org/data-structures/.

3. W3Schools, “Web Development Tutorials,” [Online]. Available: https://www.w3schools.com/.

4. FreeCodeCamp, “Tic Tac Toe Game in JavaScript,” [Online]. Available: https://www.freecodecamp.org/news/tic-tac-toe-javascript-game-tutorial/.

5. Netlify, “Docs – Deploying Websites,” [Online]. Available: https://docs.netlify.com/.

6. E. Horowitz, S. Sahni, and S. Anderson-Freed, *Fundamentals of Data Structures in C*, 2nd Edition, W.H. Freeman and Company, 1993.

7. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (CLRS)*, 3rd Edition, MIT Press, 2009.

8. Narasimha Karumanchi, *Data Structures and Algorithms Made Easy*, CareerMonk Publications, 2011.