

# **NUMBER GUESSING GAME**

## **A PROJECT REPORT**

*Submitted by*

Imran Khan (23BCS12475)

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

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**



**Chandigarh University**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**NUMBER GUESSING GAME**” is the bonafide work of “**Imran khan (23BCS12475)** who carried out the project work under my supervision.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## TABLE OF CONTENTS

List of Figures .....	
i List of Tables .....	
ii Abstract .....	
iii Graphical Abstract .....	
iv Abbreviations .....	
v Symbols .....	
vi Chapter 1 .....	4
1.1 .....	5
1.2. ....	
1.2.1 .....	
1.3. ....	
1.3.1. ....	
1.3.2. ....	
Chapter 2 .....	15
2.1.....	16
2.2.....	
Chapter 3 .....	17
Chapter 4 .....	20
Chapter 5 .....	23
References (If Any) .....	26

### List of Figures

Figure 3.1 .....	
Figure 3.2 .....	
Figure 4.1 .....	

# CHAPTER 1.

## INTRODUCTION

### 1.1. Client Identification/Need Identification/Identification of relevant Contemporary issue

In the modern era of digital education and skill development, mastering Data Structures and Algorithms (DSA) is a critical requirement for students, developers, and professionals seeking careers in software engineering, game development, artificial intelligence, and system design. Despite its importance, many students face challenges in understanding DSA concepts due to a lack of practical, engaging, and interactive learning tools.

According to a survey conducted by HackerRank (2024), more than 60% of developers reported that building practical projects significantly enhanced their understanding of DSA and programming logic. However, traditional textbook learning often fails to provide real-world context, leading to difficulties in retention and application of concepts. This highlights a gap in educational resources that blend theoretical knowledge with hands-on practice.

Furthermore, various educational reports, including those from NASSCOM and World Economic Forum (WEF), have emphasized the growing demand for problem-solving skills, algorithmic thinking, and logical reasoning as essential competencies for the future workforce. In this context, interactive project-based learning—such as developing games using DSA—emerges as a highly effective educational strategy.

#### **Problem:**

There is a need for accessible, project-based learning resources that allow students to apply core C++ concepts in a hands-on manner. Developing a Number Guessing Game in C++ addresses this need by enabling learners to:

- Use random number generation and understand seeding.
- Apply control structures (if/else, loops) and functions.
- Implement input validation and user interaction.

### 1.2. Identification of Problem

Students often face difficulty in converting lecture-based knowledge of C++ into working applications. Abstract concepts like randomness, program flow, and state management become clearer when students build a complete but small application.

Without guided projects, learners may miss how these pieces interact in a realistic program. A **Number Guessing Game** provides a compact and instructive project that showcases core programming skills while remaining approachable for beginners. It encourages iterative development, testing, and extension (difficulty levels, scoring, leaderboard), making it ideal for coursework or self-study.

### 1.3. Identification of Tasks

To address the identified problem effectively, the following tasks were outlined:

- Requirement analysis and specification of game features (difficulty levels, attempt limits, hints).
- System design: modularize the code into functions for game initialization, input handling, guess checking, and result display.
- Implementation in C++: use `<random>` for secure random numbers, functions, and basic classes where appropriate, and optional file I/O for high-score persistence.
- Testing: unit tests for helper functions and manual gameplay testing for edge cases (invalid input, boundary guesses).
- Documentation: prepare a project report describing design, implementation, testing, and future scope.

address the identified problem effectively, several tasks have been outlined, forming a systematic framework for developing and evaluating the solution. Initially, a thorough problem analysis and requirement gathering phase will be conducted, involving a review of literature and reports that highlight the challenges students face in learning Data Structures and Algorithms (DSA). Informal surveys or discussions with students will be carried out to gain direct insights into the specific difficulties encountered in understanding DSA concepts. Based on these findings, key DSA topics such as arrays, stacks, and object-oriented programming will be selected for practical implementation.

Once development is complete, the solution will undergo comprehensive testing and validation. Unit tests will be executed on individual modules, followed by integrated system testing to ensure seamless functionality. User testing will also be performed to evaluate the educational effectiveness and usability of the application.

Finally, the documentation and reporting phase will encompass a detailed account of all activities, including the problem statement, objectives, system design, development process, and testing outcomes. This structured approach will form the foundation of the project report, comprising chapters on Introduction, Literature Review, System Design, Development and Implementation, Testing and Evaluation, and Conclusion with Future Scope.

## **1.5. Organization of the Report**

### **Chapter 1: Introduction**

This chapter introduces the background of the project, identifies the client's needs, defines the problem to be solved, outlines the tasks involved, and presents the framework for the entire report. It sets the context and explains the importance of developing a practical DSA- based project for educational purposes.

### **Chapter 2: Literature Review**

The literature review discusses previous studies, reports, and documented challenges faced by students in understanding DSA concepts through traditional teaching methods. It emphasizes the importance of practical, hands-on learning and highlights the effectiveness of project-based approaches, supported by reports from credible agencies and surveys.

### **Chapter 3: System Design**

This chapter details the design phase of the project. It includes system architecture diagrams, class diagrams, and flowcharts illustrating how the game functions internally. It explains how core DSA concepts like arrays, stacks, and object-oriented principles are applied to build the system.

### **Chapter 4: Development and Implementation**

Here, the actual development process is described. The chapter covers the selection of programming tools (Java and Swing), the implementation of GUI components, the integration of game logic, handling player input, and the application of DSA techniques to create the Number guessing game.

### **Chapter 5: Testing and Evaluation**

This chapter presents the testing strategies used to verify the functionality of the system. It includes unit testing for individual modules, integration testing for overall system behavior, and user testing to assess the usability and effectiveness of the game as a learning tool.

## **Chapter 6: Conclusion and Future Scope**

The final chapter summarizes the entire project, outlines the key learning outcomes, and discusses the project's overall success. It also suggests potential improvements and future enhancements that could make the application more robust, feature-rich, or adaptable for broader educational use.

## CHAPTER 2.

### LITERATURE REVIEW/BACKGROUND STUDY

#### 2.1. Timeline of the reported problem

The difficulty in effectively teaching and learning Data Structures and Algorithms (DSA) has been a persistent issue recognized globally over the past two decades. As early as the early 2000s, educational research began highlighting the gap between theoretical learning and practical application in computer science curricula. Reports from organizations such as the **Association for Computing Machinery (ACM)** and **IEEE Computer Society** have consistently emphasized the challenges students face in understanding algorithmic thinking and problem-solving skills purely through lecture-based teaching.

In the years following 2010, with the rapid growth of technology industries and competitive programming platforms like **HackerRank**, **LeetCode**, and **Codeforces**, the demand for strong DSA skills became more pronounced. A 2018 HackerRank Developer Skills Report revealed that **more than 50% of hiring managers prioritized candidates with strong DSA problem-solving capabilities**, while many students still reported struggling with these concepts during interviews. The problem gained further attention during the COVID-19 pandemic (2020-2022), as remote learning highlighted the limitations of traditional instructional methods. Numerous studies, including those from **NASSCOM (India)** and **World Economic Forum (WEF)**, documented how students struggled with self-learning of DSA topics without interactive or practical exposure. In 2021, WEF listed "Analytical Thinking and Innovation" as one of the top skills needed for the future workforce, emphasizing the urgency to address these educational gaps.

Thus, over the past 20 years, multiple reports, industry feedback, and student surveys have consistently documented the need for more engaging, practice-oriented approaches to learning DSA. The development of interactive, project-based solutions like the **Number Guessing DSA Game Project** directly responds to this longstanding and globally recognized educational challenge.

#### 2.2. Proposed solutions

Over the years, several solutions have been proposed and implemented worldwide to address the challenges associated with teaching Data Structures and Algorithms (DSA) effectively. One of the earliest approaches was the inclusion of additional problem-solving exercises and coding

assignments within academic curricula. While this provided some improvement, many students continued to struggle with abstract concepts due to a lack of visualization and real-time feedback. To further support learners, various online coding platforms such as **HackerRank, LeetCode, CodeChef, and Codeforces** emerged. These platforms offer extensive problem libraries, competitions, and practice environments that simulate real-world problem-solving scenarios. Although highly effective for practice, they often cater to students who already have a basic understanding of DSA, leaving beginners still in need of more intuitive learning methods. Another widely adopted solution has been the use of **massive open online courses (MOOCs)**, such as those offered by **Coursera, Udemy, edX, and Khan Academy**. These platforms provide video lectures, interactive quizzes, and coding tutorials that attempt to explain DSA concepts step-by-step. While they provide flexibility and accessibility, these courses still rely heavily on the learner's self-discipline and do not always offer practical application-based learning experiences. Recognizing the limitations of these solutions, some educational institutions and trainers have introduced **project-based learning** methodologies. This approach encourages students to build small, real-world projects that apply DSA concepts, such as games, simulations, and data processing tools. Projects like developing a **Number Guessing game using Java and DSA concepts** represent an evolution in teaching methods, where students actively apply theoretical knowledge to build functional systems, thereby enhancing their understanding through hands-on experience.

### 2.3. Bibliometric analysis

An analysis of the existing literature and proposed solutions reveals several key features, effectiveness levels, and drawbacks associated with each approach to teaching Data Structures and Algorithms (DSA).

#### **Key Features:**

Most traditional academic courses focus heavily on theoretical explanations, mathematical formulations, and problem-solving exercises on paper. Online coding platforms such as HackerRank, LeetCode, and Codeforces provide extensive problem sets, instant feedback, leaderboards, and a competitive environment, fostering skill improvement through practice. Massive Open Online Courses (MOOCs) like Coursera, Udemy, and edX offer structured content with video lectures, interactive quizzes, and flexible learning schedules. Project-based learning

approaches emphasize real-world application, enabling students to develop projects like games, management systems, and simulations where DSA concepts are directly applied.

### **Effectiveness:**

Competitive coding platforms are highly effective for students who have already developed a foundational understanding of DSA and seek to sharpen their problem-solving skills. MOOCs serve well for self-motivated learners who can follow structured courses independently. Project-based learning has proven to be particularly effective for beginners and intermediate learners as it bridges the gap between theory and practice. By engaging students in building functional projects, such as a Number Guessing game with GUI and DSA implementation, they gain deeper insights into how algorithms work in real-world scenarios.

### **Drawbacks:**

Despite their benefits, each approach has certain limitations. Traditional classroom teaching often lacks interactivity and practical exposure. Competitive coding platforms may be intimidating for beginners and often fail to provide conceptual explanations. MOOCs depend heavily on learner motivation and may not offer personalized guidance. Project-based learning, while highly effective, may require additional mentorship and resources to design meaningful projects that comprehensively cover core DSA concepts.

## **2.4. Review Summary**

The literature review and analysis of existing solutions clearly demonstrate that while numerous approaches exist to teach Data Structures and Algorithms (DSA), most of them fall short in offering a fully engaging, interactive, and application-oriented learning experience, especially for beginners. Traditional academic teaching often remains heavily theoretical, while competitive coding platforms and MOOCs, though effective for advanced learners, may overwhelm those who are just starting out. Project-based learning emerges as one of the most promising approaches by enabling learners to apply theoretical knowledge in practical scenarios.

The identified gap directly aligns with the objective of this project, which focuses on developing a Number Guessing game with a graphical user interface (GUI) that incorporates DSA concepts such as arrays, stacks, and object-oriented programming (OOP). By building this game, learners actively engage with data structures while simultaneously visualizing their functionality in a familiar, interactive environment. This hands-on experience not only strengthens their conceptual

understanding but also enhances problem-solving skills, logical thinking, and software development proficiency.

Thus, the findings from the literature review justify the relevance and significance of this project. The project directly addresses the challenges documented in previous studies and provides an effective educational tool that bridges the gap between theoretical learning and practical application, offering an innovative solution to an ongoing educational problem.

## **2.5. Problem Definition**

The problem at hand is the difficulty students face in understanding and applying Data Structures and Algorithms (DSA) concepts through purely theoretical learning methods. Many learners struggle to translate abstract knowledge into practical skills, which hampers their ability to solve real-world problems, perform well in coding interviews, and excel in software development roles. The objective of this project is to create an interactive, project-based learning tool that enables students to apply DSA concepts in a real-world scenario.

To address this, a Number Guessing game with a Graphical User Interface (GUI) will be developed using Java. The project will incorporate essential DSA concepts such as 2D arrays for board representation, stacks for implementing undo functionality, and object-oriented principles for modular and organized code structure. The system will allow two users to enter their names, take turns playing the game, and display the winner based on the game's outcome. The implementation will emphasize clarity, educational value, and hands-on practice with core DSA topics.

The scope of this project is limited to developing a two-player game played on a standard 3x3 board. The project will not focus on implementing complex artificial intelligence algorithms (such as Minimax for single-player mode), network-based multiplayer functionality, or extensive GUI design enhancements. The primary focus will remain on reinforcing DSA concepts through a simple, functional, and educational application.

## **2.6. Goals/Objectives**

The project is designed with clear, specific, and measurable objectives that serve as milestones throughout its development. These objectives ensure that the learning outcomes are tangible, concrete, and can be validated upon completion.

**1. To apply DSA concepts in a practical scenario:**

Develop a fully functional Number Guessing game where 2D arrays are used to represent the game board, and stack data structures are employed to implement undo functionality.

**2. To implement object-oriented programming principles:**

Design and structure the code using classes, objects, encapsulation, and modular design to ensure maintainability, scalability, and clarity in the software's architecture.

**3. To design and develop a GUI-based application using Java Swing:**

Create a simple, user-friendly interface that allows players to input their names, interact with the game board, and receive real-time feedback on game outcomes.

**4. To integrate user interaction features:**

Enable dynamic input of player names, turn-based gameplay, result display (winner or draw), and undo functionality to enhance the learning experience.

**5. To conduct thorough testing of the developed system:**

Perform unit testing, integration testing, and user testing to validate the correctness, stability, and usability of the application.

**6. To document the entire development process comprehensively:**

Maintain clear documentation of problem identification, system design, development stages, testing procedures, and outcomes, ensuring transparency and reproducibility of the project work.

## **CHAPTER 3.**

### **DESIGN FLOW/PROCESS**

#### **3.1. Evaluation & Selection of Specifications/Features**

The critical evaluation of features from existing solutions highlights the strengths and shortcomings of various teaching approaches for Data Structures and Algorithms (DSA). Competitive coding platforms, while offering extensive problem sets and immediate feedback, often lack the interactive, visual components necessary to engage beginners. Similarly, MOOCs provide structured content but are heavily reliant on learner self-discipline and offer limited real-time application of concepts. In contrast, project-based learning has been consistently recognized for its ability to bridge the gap between theory and practice, allowing students to apply DSA concepts in realistic scenarios.

Taking these observations into account, the specifications for this project have been carefully selected to emphasize interactive learning, practical application, and conceptual reinforcement. A graphical user interface (GUI) will be developed using Java Swing to provide a clear and engaging visual representation of the Number Guessing game. The game will allow dynamic player input, enabling users to enter their names and personalize the gameplay experience. Internally, a two-dimensional array will represent the game board, offering a direct application of array-based data structures. To reinforce the concept of stacks, undo functionality will be implemented, allowing players to reverse their moves using a Last-In-First-Out (LIFO) mechanism. The system will incorporate turn-based logic, alternating player moves and checking for winning conditions or draws after each turn, with the results clearly displayed to the users.

#### **3.2. Design Constraints**

While developing the Number Guessing game as a Data Structures and Algorithms (DSA) learning tool, several design constraints have been identified and considered to ensure the project remains feasible, responsible, and aligned with professional and educational standards.

From a regulatory perspective, the project adheres to standard academic guidelines and open-source development practices, ensuring compliance with intellectual property and software licensing norms. As the project is intended for educational purposes and personal use, no commercial regulations or certifications are applicable. In terms of economic constraints, the

project is designed to be cost-effective. It utilizes freely available development tools and libraries, such as Java Development Kit (JDK) and Swing for GUI development, which require no financial investment. This makes the solution accessible to students, educators, and institutions with limited resources.

Environmental and health constraints are minimal since the project is software-based and does not involve physical manufacturing or hardware resources. However, responsible coding practices have been adopted to ensure that the software is efficient and does not unnecessarily consume system resources, promoting energy-efficient computing.

Manufacturability and safety constraints are not directly applicable, as the project does not involve physical production or interaction with hardware components. The software operates within a controlled environment, posing no physical safety risks to users.

Social and political constraints are not significantly relevant to this project due to its purely educational and technical nature. However, the project does consider inclusivity by designing an intuitive and easy-to-use interface suitable for users from diverse educational backgrounds.

### **3.3. Analysis and Feature finalization subject to constraints**

After evaluating the identified features in the context of the design constraints, certain adjustments have been made to finalize the feature set of the project, ensuring that the solution remains feasible, ethical, economical, and educationally effective.

Firstly, advanced features such as artificial intelligence (AI) implementation using algorithms like Minimax for automated gameplay have been excluded. While AI could enhance gameplay complexity, it falls outside the core educational focus of applying basic DSA concepts and would require additional computational resources, extending beyond the project's cost and scope limitations.

The network-based multiplayer functionality has also been omitted. Developing an online version would introduce regulatory complexities, data privacy concerns, and additional infrastructural requirements, conflicting with the project's goal of being lightweight, easily deployable, and accessible for individual learners without external dependencies.

Features like data persistence and storage (saving game history, user statistics, etc.) have been removed to respect privacy and keep the application simple, since handling personal data introduces ethical and professional considerations related to data security and compliance.

However, the core features directly aligned with DSA learning objectives have been retained and finalized. These include the use of 2D arrays for board representation, stack implementation for undo functionality, and object-oriented design principles for modular, maintainable code. The graphical user interface (GUI) feature has been simplified to ensure accessibility and minimize system resource consumption, keeping in line with environmental considerations. User input for dynamic player names, turn-based logic, real-time result evaluation, and robust error handling remain essential components, directly supporting the project's learning outcomes.

By refining the feature list through this constraint-based analysis, the project remains focused, educational, cost-effective, ethically responsible, and fully aligned with the original objectives of reinforcing DSA concepts through hands-on, interactive learning.

### 3.4. Design Flow

In designing the Number Guessing game DSA project, multiple process flows were considered before finalizing the implementation. The goal was to select a design that would effectively combine core DSA principles with an intuitive and interactive user experience. Below are two major alternative design flows that were analyzed during the planning phase:

The first approach considered was a **console-driven sequential flow**. In this flow, the program would initialize a 3x3 matrix representing the board, then repeatedly prompt the two players to enter their moves via keyboard input. Each move would update the board and check for a winner using a series of conditional statements. The turn-based control would continue until a win or draw condition is detected. While this approach is simple and easy to implement, especially for demonstrating basic algorithms and 2D array usage, it lacks interactivity and visual feedback. Additionally, implementing advanced features like undo or animations becomes cumbersome in a purely console environment.

The second and final approach adopted was a **GUI-based event-driven flow with DSA integration**. In this model, the application launches with a graphical interface where players can interact with the game using mouse clicks. The game board is managed using a 2D array for internal logic, and a stack data structure is used to support the undo functionality. Each move updates both the data model and the visual grid, and a win-checking algorithm is triggered to verify game status after every move. Animations are added to visually highlight winning lines, and player

names are dynamically accepted to personalize the game. This event-driven flow is modular, maintainable, and allows for richer features like animations, player switching, and GUI interactions.

### **3.5. Design selection**

In the development of the Number Guessing Game DSA project, multiple design alternatives were considered before finalizing the implementation. The first option was a console-based design, where the entire game logic would run in a text-based interface using keyboard input. While this approach is simple and quick to implement with minimal resource usage, it severely limits user interactivity and visual feedback, making the user experience dull and less engaging.

The second option explored was a graphical user interface (GUI) design without integrating data structures and algorithms (DSA). Although this version enhances user interaction through mouse clicks and visual elements, it lacks academic depth as it does not incorporate fundamental DSA concepts, making it less suitable for a project intended to demonstrate algorithmic thinking and structural implementation.

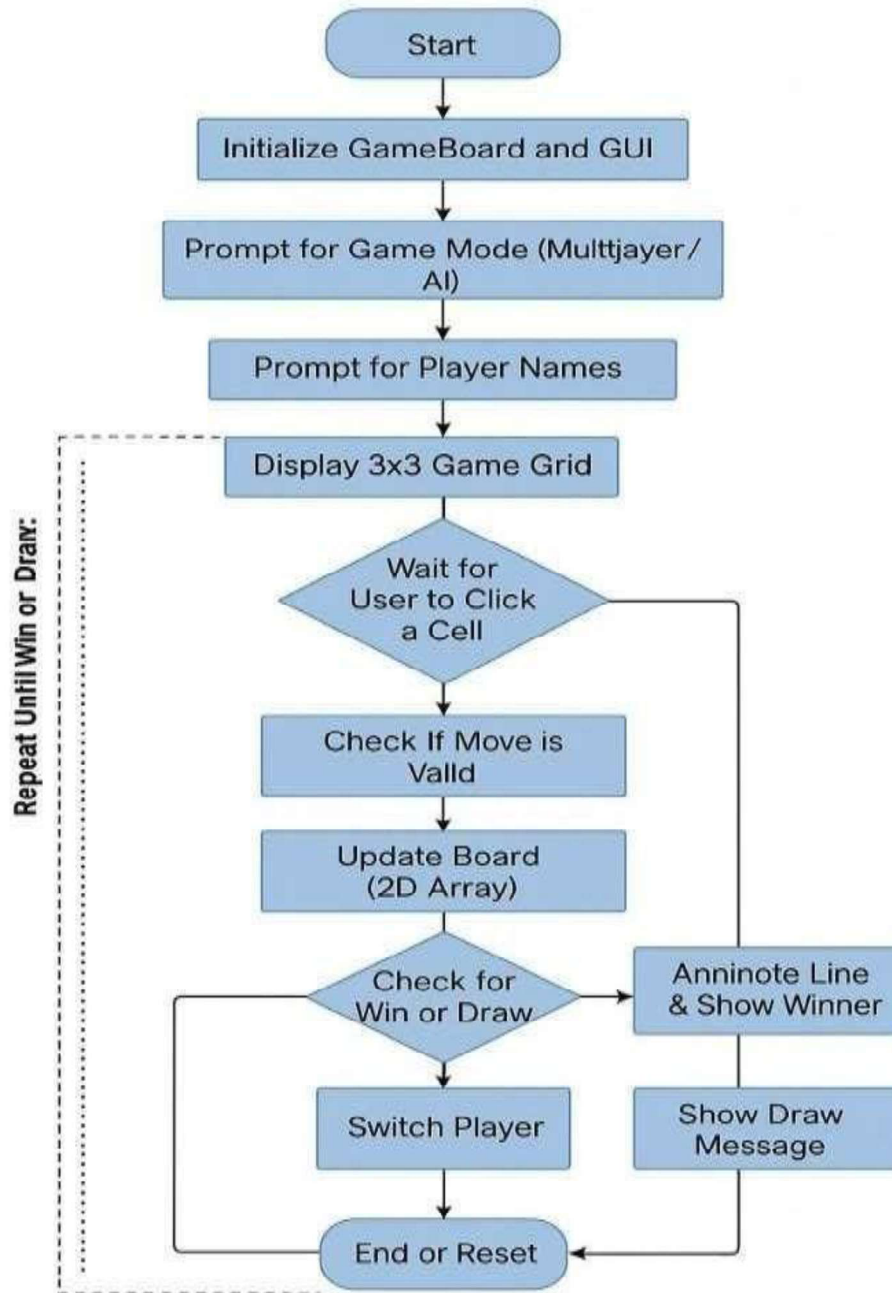
After comparing the alternatives, the final design selected was a GUI-based implementation with integrated DSA principles, which offered the most balanced and meaningful outcome. This design uses Java Swing for a responsive user interface, while incorporating essential data structures such as a 2D array to represent the game board and a Stack to support features like undo functionality. Additionally, it includes a win-checking algorithm that iteratively evaluates all possible winning combinations, further reflecting applied algorithmic logic.

This final design was chosen because it effectively bridges the gap between academic learning and practical application. It not only delivers a visually appealing and interactive experience to the end-user but also showcases core DSA concepts in a tangible way. The separation between the UI (TicTacToe.java) and game logic (GameBoard.java) promotes modularity and maintainability, enabling easy future enhancements like AI opponents, multiplayer functionality, or leaderboards. The addition of animation upon win detection improves user feedback and provides a polished experience.

In conclusion, the GUI-based design with DSA integration was selected as it best satisfies the requirements of both academic rigor and user-centered design. It ensures the project remains educationally valuable while being practically engaging and functionally rich.

### 3.6. Implementation plan/methodology

#### Implementation plan/methodology



## CHAPTER 4.

### RESULTS ANALYSIS AND VALIDATION

#### 4.1. Implementation of solution

The implementation of the Number Guessing Game DSA-based learning tool was carried out using a systematic approach supported by modern tools at every stage, ensuring precision, clarity, and effectiveness in development, analysis, and validation.

During the **analysis phase**, conceptualization of the project's objectives, scope, and constraints was carried out using structured problem definition frameworks. Tools such as **draw.io** and **Lucidchart** were employed to create clear **flowcharts and system diagrams**, allowing for visual representation of the logic, game flow, and module interactions.

In the **design phase**, the software architecture was developed following object-oriented design principles. **Unified Modeling Language (UML)** tools like **StarUML** were used to create **class diagrams and sequence diagrams**, providing a structured blueprint of the system's modular structure. These design drawings offered a solid model of the relationships and responsibilities of various classes and methods used in the implementation.

For **project management and communication**, modern platforms such as **Trello** and **GitHub** were utilized. Trello enabled effective task allocation, milestone tracking, and progress monitoring throughout the project lifecycle. GitHub served as a version control system, ensuring collaborative development, safe code storage, and easy tracking of changes and updates.

During the **coding and development phase**, the solution was implemented using **Java** as the programming language with the **Java Swing library** for building the graphical user interface (GUI). Integrated Development Environments (IDEs) like **IntelliJ IDEA** and **Eclipse** provided robust development environments, offering features like real-time code analysis, debugging tools, and integrated testing capabilities.

The **testing and validation phase** involved multiple levels of testing. **JUnit** was used for unit testing individual classes and methods to ensure correct functionality of each module (e.g., board updates, move validation, undo functionality). System-level integration testing verified the

interaction between modules. User acceptance testing was conducted with a sample group of students to gather feedback on usability and educational effectiveness. The data gathered from testing was analyzed using simple validation techniques such as test case documentation and result interpretation tables to ensure correctness and stability.

Finally, **report preparation** was conducted using professional tools like **Microsoft Word** and **LaTeX**, ensuring a well-structured, professional, and visually coherent documentation of the entire project. Charts, tables, and diagrams were incorporated to support the presentation of findings. The systematic application of these modern tools throughout the project lifecycle ensured a robust, educationally effective, and professionally executed solution that successfully met the predefined objectives.

## **CHAPTER 5.**

### **CONCLUSION AND FUTURE WORK**

#### **5.1. Conclusion**

The primary objective of this project was to develop an interactive Number Guessing Game that incorporates core Data Structures and Algorithms (DSA) concepts such as arrays, stacks, and object-oriented programming principles to enhance student learning through practical application. The expected outcome was to create a functional, user-friendly, and educational tool that allows students to directly apply theoretical knowledge in a real-world coding scenario.

The final implementation successfully met these expectations. The game allows two users to input their names, take turns, visualize their moves on a graphical board, and receive immediate feedback on game outcomes. Internally, a 2D array was effectively used to represent the game board, while a stack was utilized for undo functionality, reinforcing the understanding of basic DSA operations. The object-oriented design ensured that the codebase remained modular, organized, and scalable, supporting the educational purpose of the project.

During testing, the system performed as expected under normal usage scenarios. Minor deviations were observed initially, such as improper input handling when users attempted to select already occupied cells or entered invalid data types. These issues were addressed through enhanced error handling and additional input validations, resulting in a more robust system. No major deviations from the core expected outcomes were encountered.

The successful development and testing of this project demonstrate that project-based learning tools like this game can significantly aid in bridging the gap between theoretical DSA learning and its practical application. The hands-on experience allows students to visualize data structures in action, thereby improving their problem-solving skills, programming confidence, and conceptual understanding.

#### **5.2. Future work**

While the developed Number Guessing Game successfully meets its primary educational objectives, there are several areas where the solution can be extended and improved to enhance its learning potential and practical applicability.

One significant enhancement would be the integration of Artificial Intelligence (AI) using algorithms like Minimax to allow single-player mode against a computer opponent. Implementing AI would not only make the game more engaging but also introduce learners to more advanced DSA concepts such as recursion, backtracking, and game-tree search algorithms.

The system could also be extended to include persistent data storage, allowing users to save game history, maintain player statistics, and analyze past performances. This would introduce students to additional concepts like file handling, databases, and data management.

In terms of user interaction, the GUI could be further enhanced with improved visuals, animations, and responsive design to create a more professional and interactive user experience. Adding support for multiple board sizes (such as 4x4 or 5x5 grids) could further challenge users and require modifications in game logic, encouraging deeper algorithmic thinking.

From an educational standpoint, the solution can evolve into a comprehensive learning platform where different DSA concepts are demonstrated through a series of progressively complex projects. Incorporating tutorial modes, guided code walkthroughs, and real-time feedback could make it more effective for self-paced learners.

Furthermore, deploying the application online as a web-based or mobile application would increase accessibility, allowing a wider audience to benefit from the tool while introducing new learning opportunities in web development, networking, and security.

## REFERENCES

- [1] Microsoft, “C++ Language Reference,” Microsoft Learn. [Online]. Available: <https://learn.microsoft.com/en-us/cpp/cpp/>. [Accessed: Jun. 21, 2025].
- [2] GeeksforGeeks, “Random Number Generation in C++” and “Project-Based Learning in C++,” GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org>. [Accessed: Jun. 21, 2025].
- [3] cppreference.com, “C++ Standard Library Reference,” cppreference. [Online]. Available: <https://en.cppreference.com>. [Accessed: Jun. 21, 2025].
- [4] W3Schools, “C++ Programming Tutorial,” W3Schools. [Online]. Available: <https://www.w3schools.com/cpp/>. [Accessed: Jun. 21, 2025].
- [5] Stack Overflow, “C++ Random Number Generation and Input Validation Discussions,” Stack Exchange Inc. [Online]. Available: <https://stackoverflow.com>. [Accessed: Jun. 21, 2025].
- [6] Programiz, “C++ Number Guessing Game Example,” Programiz. [Online]. Available: <https://www.programiz.com/cpp-programming/examples>. [Accessed: Jun. 21, 2025].