TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A Proposal on
Developing a Sudoku Game Using Object-Oriented Programming

Submitted By:

Saurav Gaudel (PUL081BCT076)
Sudip Shrestha (PUL081BCT084)
Yogendra Sharma Upadhyaya (PUL081BCT095)

Submitted To:

Daya Sagar Baral

Department of Electronics and Computer Engineering

Pulchowk Campus
Kathmandu, Nepal

July 2025

ACKNOWLEDGEMENT

We sincerely appreciate the Institute of Engineering, Pulchowk Campus, for providing us with this project as an opportunity to advance our skills and knowledge. We sincerely thank the Department of Electronics and Computer Engineering for their continuous support and for giving us access to vital data and resources. We especially thank our OOP Teacher Daya Baral Sir, for his invaluable advice, assistance, and knowledgeable insights during this study. The practical application and educational experience have been immensely beneficial, and we are grateful to everyone who helped us along the way.

Saurav Gaudel (PUL081BCT076)
Sudip Shrestha (PUL081BCT084)
Yogendra Sharma Upadhyaya (PUL081BCT095)

Table of Content

1. INTRODUCTION

Sudoku is a popular number-placement puzzle consisting of a 9x9 grid divided into smaller 3x3 boxes. The goal is to fill the grid so that each row, column, and box contains the numbers 1 through 9 without repetition. Despite its simple rules, Sudoku puzzles vary widely in difficulty, making them excellent tools for developing logical thinking and problem-solving skills.

This project aims to develop a Sudoku game application using Object-Oriented Programming (OOP) principles. By designing the system with classes representing the board, cells, and solver, the project will demonstrate important OOP concepts such as encapsulation and modularity. Additionally, the application will include a solver capable of automatically completing puzzles using algorithmic techniques like backtracking.

While many Sudoku apps exist, this project focuses on creating a clean, maintainable design that is easy to extend and debug. The system will provide an interactive user interface to allow users to play, validate, and solve Sudoku puzzles.

The scope of this project is limited to desktop gameplay and solving features, without online or mobile components. This proposal outlines the design, objectives, and methodology for building a functional Sudoku game that combines algorithmic logic with software engineering best practices.

1.1 Background

As first-year Bachelor in Computer Engineering students, we begin learning OOP in our II semester course to develop our logical reasoning and problem-solving abilities. Our goal for the semester project was to use basic OOP concepts like object, classes, inheritance, encapsulation, polymorphism, and arrays to create a simple Sudoku game.

1.2 History

Although widely associated with Japanese culture, Sudoku did not originate in Japan. The roots of Sudoku can be traced back to a French puzzle from the late 18th century, which involved placing numbers in a grid without repeating them in rows or columns. However, the modern version of Sudoku, as we know it today, was first introduced in 1979 by American architect Howard Garns under the name “Number Place” in *Dell Puzzle Magazine*.

The game was later introduced in Japan by the puzzle company Nikoli in 1984, where it was renamed "Sudoku," an abbreviation of the Japanese phrase *"Sūji wa dokushin ni kagiru"*, meaning "the digits must be single." It quickly gained popularity across Japan due to its simple rules and challenging logic.

Sudoku reached international fame in the early 2000s, largely due to its inclusion in newspapers and online puzzle platforms. The puzzle's appeal lies in its language-free logic, accessibility to players of all ages, and scalability in difficulty. Since then, it has inspired a wide variety of digital implementations and remains one of the most recognized and played logic games worldwide.

1.3 Problem Definition

Despite the widespread popularity of Sudoku as a logic-based puzzle, many existing digital implementations either lack interactive features, rely on hardcoded puzzles, or do not demonstrate clean software design practices. Additionally, many applications are built with a focus solely on gameplay, without emphasizing the underlying structure and reusability of the code.

There is a need for a Sudoku game application that is not only functional and user-friendly but also well-structured from a programming perspective. Such an application should allow users to play, input custom puzzles, and solve them automatically using an efficient algorithm. Furthermore, the system should be designed using Object-Oriented Programming (OOP) principles to ensure modularity, maintainability, and scalability.

This project aims to address these issues by developing a Sudoku game that includes:

- A user-friendly interface for puzzle input and gameplay,
- An automated solving algorithm (e.g., backtracking),
- A modular class-based structure that demonstrates core OOP concepts.

The end goal is to create an educational yet practical application that bridges algorithm design and structured software development.

2. OBJECTIVE

The main objectives of our project are listed below:

- To design and implement a Sudoku game using Object-Oriented Programming principles such as encapsulation, abstraction, and modularity.
- To develop a functional and user-friendly interface that allows users to input, play, and solve Sudoku puzzles.
- To implement an efficient Sudoku-solving algorithm (e.g., backtracking) capable of automatically solving valid puzzles.
- To demonstrate clean, maintainable, and reusable code architecture suitable for further enhancements or feature additions.

3. EXISTING SYSTEM

Several Sudoku game applications and solvers are already available across different platforms, including mobile apps, web-based games, and desktop software. These systems typically allow users to play Sudoku puzzles of varying difficulty levels and often include features such as hint systems, timer functions, and puzzle generators.

Some advanced applications also include automatic solvers that use algorithms like backtracking or constraint propagation to solve puzzles. Online platforms like Sudoku.com, mobile apps, and open-source Sudoku solvers on GitHub provide examples of such systems.

However, many of these existing systems are developed with limited focus on structured software design, making them difficult to understand, modify, or extend. In particular, beginner-level implementations may lack proper use of Object-Oriented Programming (OOP) principles, leading to tightly coupled or repetitive code.

This project aims to improve upon such limitations by building a Sudoku game with a clear and modular OOP-based structure, focusing not only on functionality but also on code maintainability and readability.

4. PROPOSED SYSTEM

The proposed system is a **graphics-based Sudoku game** developed using **C++** with the application of **Object-Oriented Programming (OOP)** principles. Unlike console-based Sudoku implementations, this system provides a more interactive and user-friendly environment where users can visually play, input, and solve Sudoku puzzles. The system will not only function as an entertaining logic game but will also serve as a demonstration of good software design practices using C++ and graphical programming techniques.

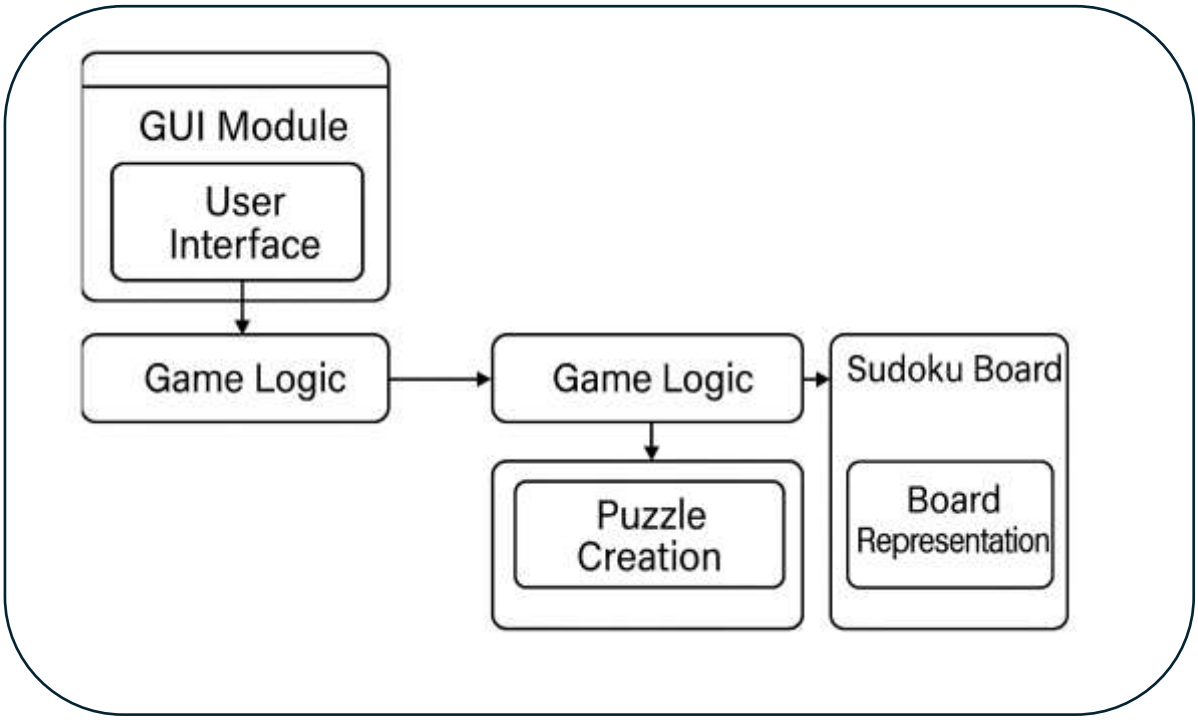
4.1 Features:

1. **Graphical User Interface (GUI)**
   The game features a GUI with a 9x9 grid displayed using graphics, allowing users to interact via mouse or keyboard.
2. **Interactive Cell Selection**
   Users can click on any cell to select it and enter digits (1–9) directly using the keyboard.
3. **Real-Time Rule Validation**
   Invalid moves are detected instantly — the system ensures no repetition of digits in rows, columns, or 3x3 subgrids.
4. **Highlighting and Visual Feedback**
   - Selected cell highlighting
   - Wrong entry indication (e.g., red color)
   - Completed row/column/box feedback (e.g., green color

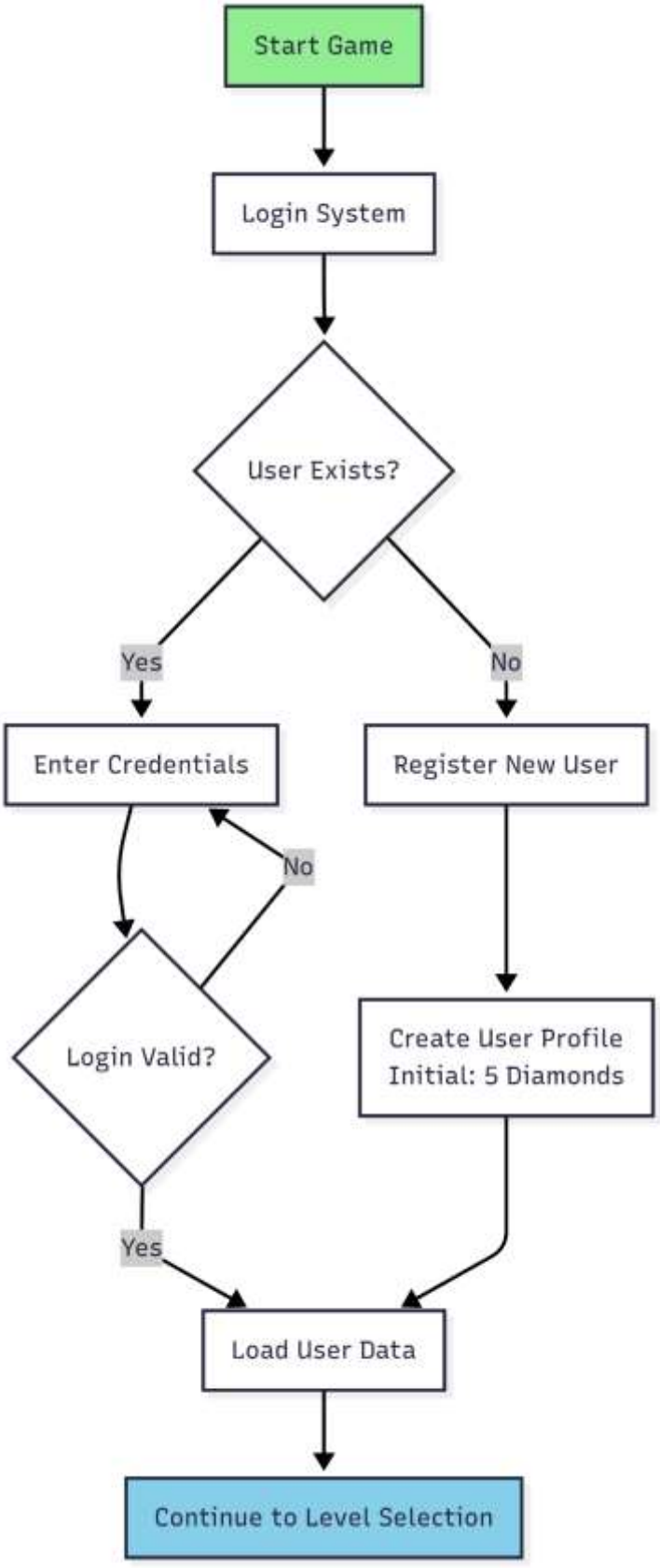
- A user-friendly interface (console-based or graphical, depending on scope).

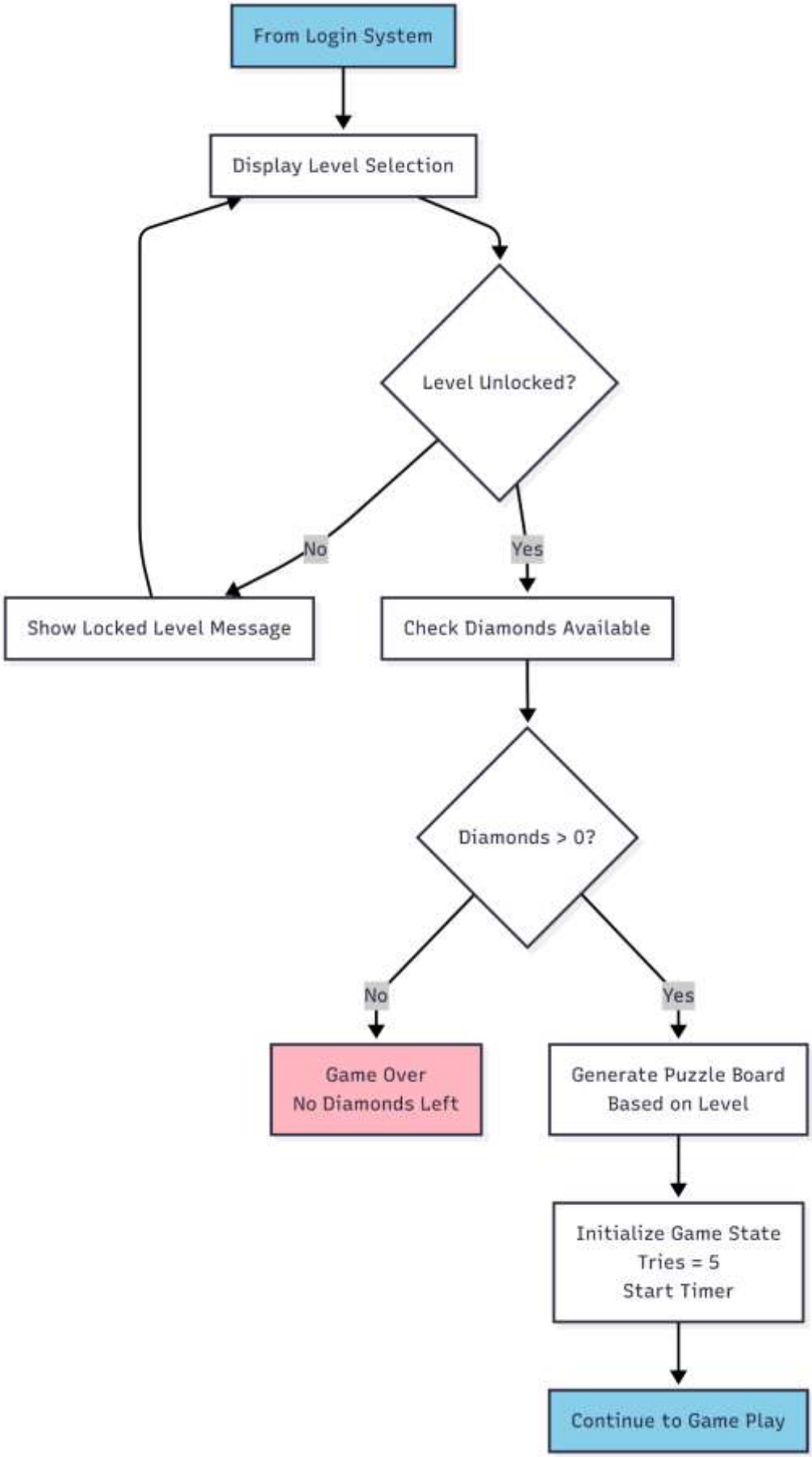
This project aims not only to deliver a working Sudoku game but also to highlight the importance of clean software design, making the codebase suitable for future feature additions such as difficulty levels, puzzle generators, or graphical interfaces.

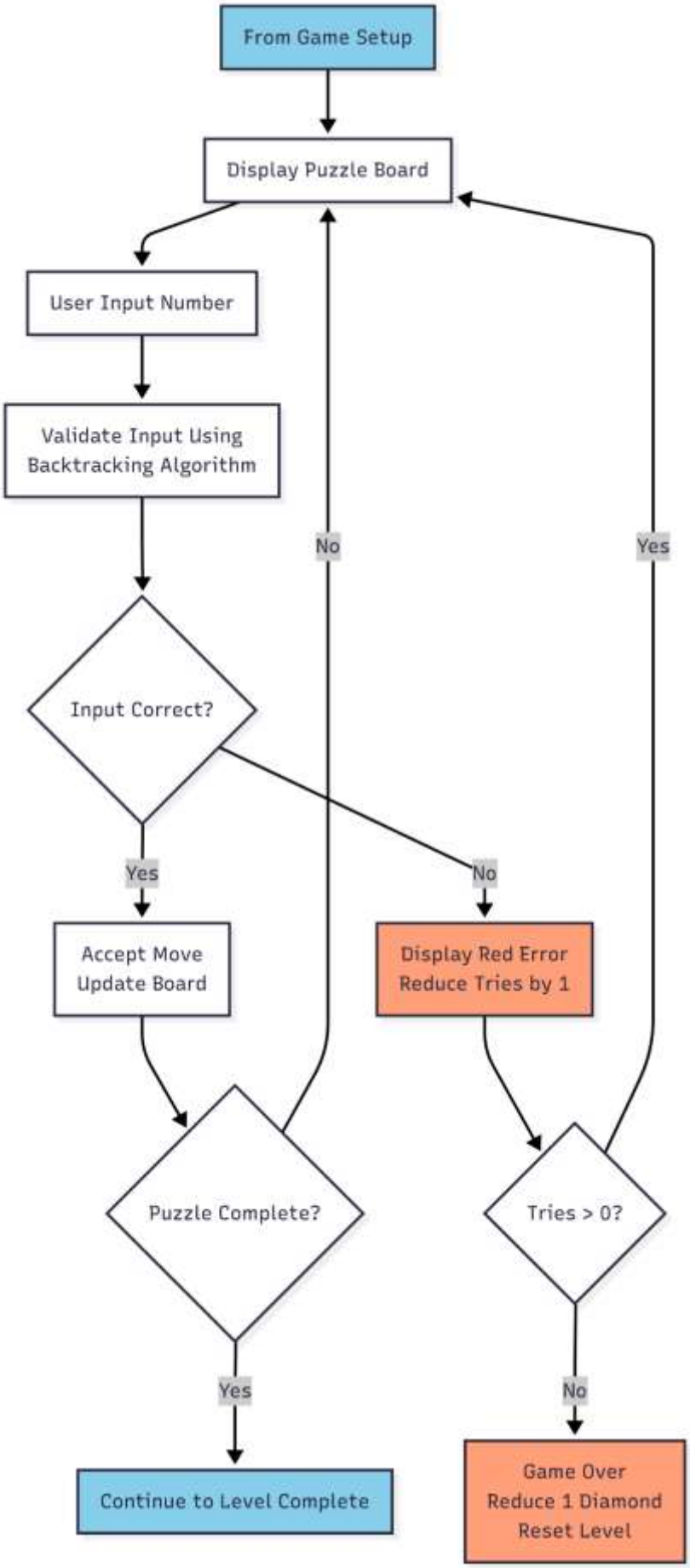
4.2 Block Diagram

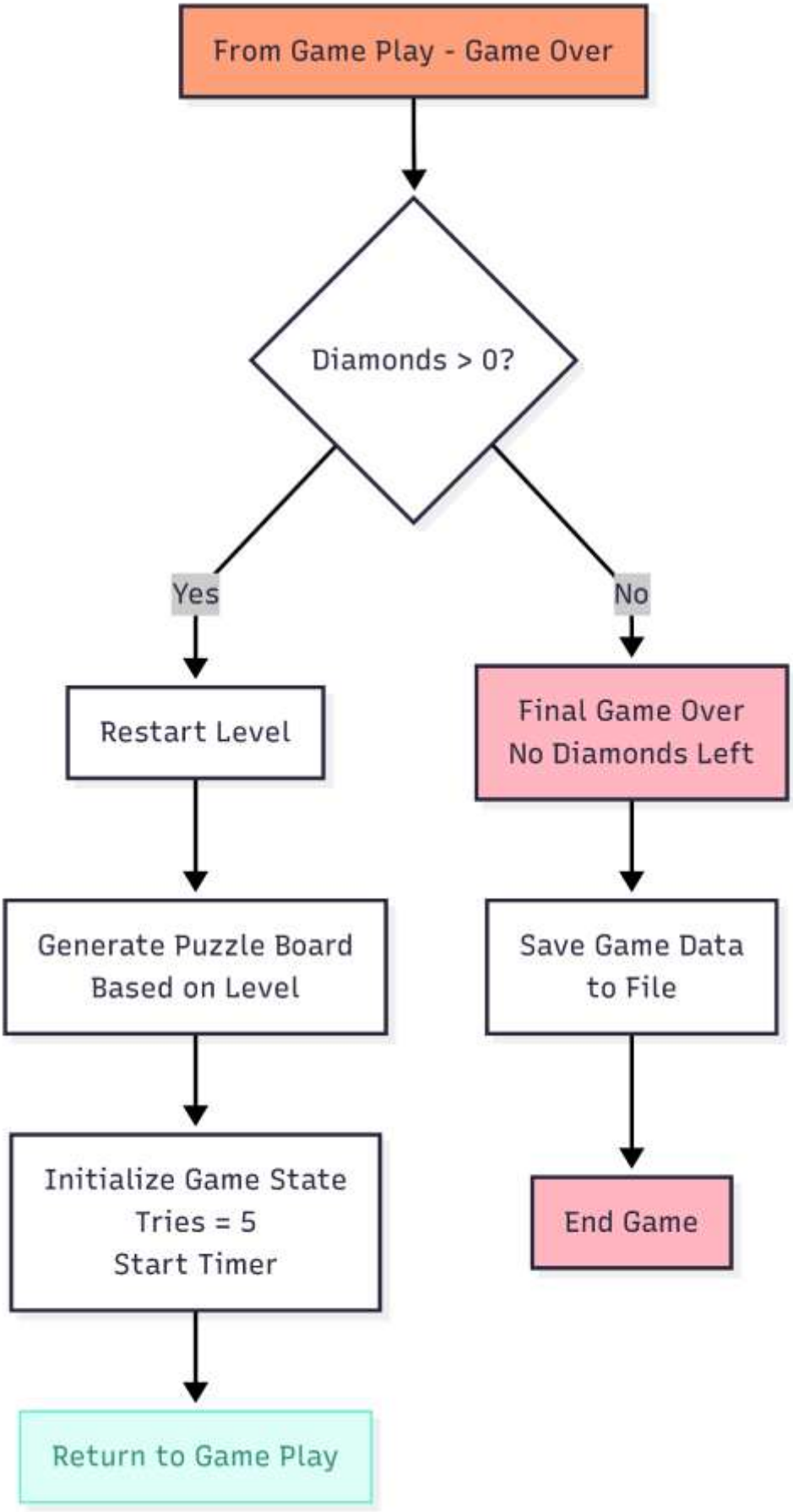


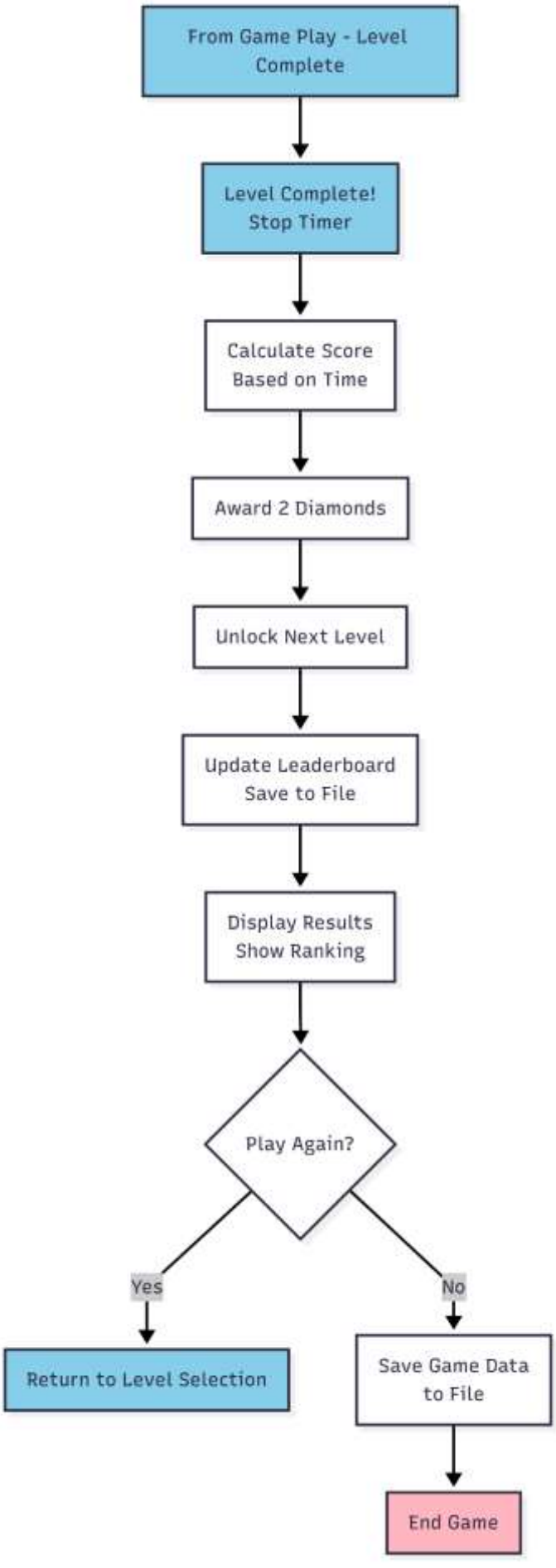
4.3 Flowchart




From Login System
Display Level Selection
Level Unlocked?
No
Yes
Show Locked Level Message
Check Diamonds Available
Diamonds > 0?
No
Yes
Game Over
No Diamonds Left
Generate Puzzle Board
Based on Level
Initialize Game State
Tries = 5
Start Timer
Continue to Game Play


From Game Setup
Display Puzzle Board
User Input Number
Validate Input Using Backtracking Algorithm
Input Correct?
No
Yes
Yes
No
Accept Move Update Board
Display Red Error Reduce Tries by 1
Puzzle Complete?
Tries > 0?
Yes
No
Continue to Level Complete
Game Over Reduce 1 Diamond Reset Level


From Game Play - Game Over
Diamonds > 0?
Yes
No
Restart Level
Final Game Over
No Diamonds Left
Generate Puzzle Board
Based on Level
Save Game Data
to File
Initialize Game State
Tries = 5
Start Timer
End Game
Return to Game Play


From Game Play - Level Complete
Level Complete! Stop Timer
Calculate Score Based on Time
Award 2 Diamonds
Unlock Next Level
Update Leaderboard Save to File
Display Results Show Ranking
Play Again?
Yes
No
Return to Level Selection
Save Game Data to File
End Game

5. METHODOLOGY

The development of the Sudoku game will follow a structured approach based on the principles of Object-Oriented Programming and graphical user interface programming using C++. The methodology includes the following stages:

1. **Requirement Analysis**

   Identify the essential features of a Sudoku game including user input, puzzle validation, solving algorithm, and GUI components (grid display, buttons, etc.).
2. **System Design**

   Use modular class-based design to separate responsibilities such as:Board class for managing grid data,
   - Cell class for handling individual cells,
   - Solver class for puzzle-solving logic,
   - UI or Game class for managing graphics and interaction.
3. **Interface Development**

   Design a graphical interface using a C++ graphics library SFML to visually represent the Sudoku grid and allow user interaction via mouse and keyboard.
4. **Algorithm Implementation**

   Implement a backtracking-based Sudoku solver that can automatically solve puzzles and validate user moves in real time.
5. **Integration and Testing**

   Integrate all modules and test for correctness, usability, and edge cases. Ensure that invalid entries are handled properly and that the interface is responsive.

6. PROJECT SCOPE

The scope of this project is to develop a fully functional, interactive, and visually intuitive Sudoku game using C++ and OOP principles. Key aspects of the project scope include:

- Development of a 9x9 Sudoku game with a graphical interface.
- Implementation of real-time user input validation.
- A built-in solver algorithm using backtracking or another efficient method.
- Use of Object-Oriented Programming to ensure modularity, reusability, and maintainability.
- Designing a simple, clean user interface using graphics.
- Scope limited to desktop environment no mobile/web version at this stage.
- Project deliverables include:
    - Source code
    - Executable program
    - Project report/proposal
    - (Optional) Flowchart/Class Diagram

7. ESTIMATED TIME SCHEDULE

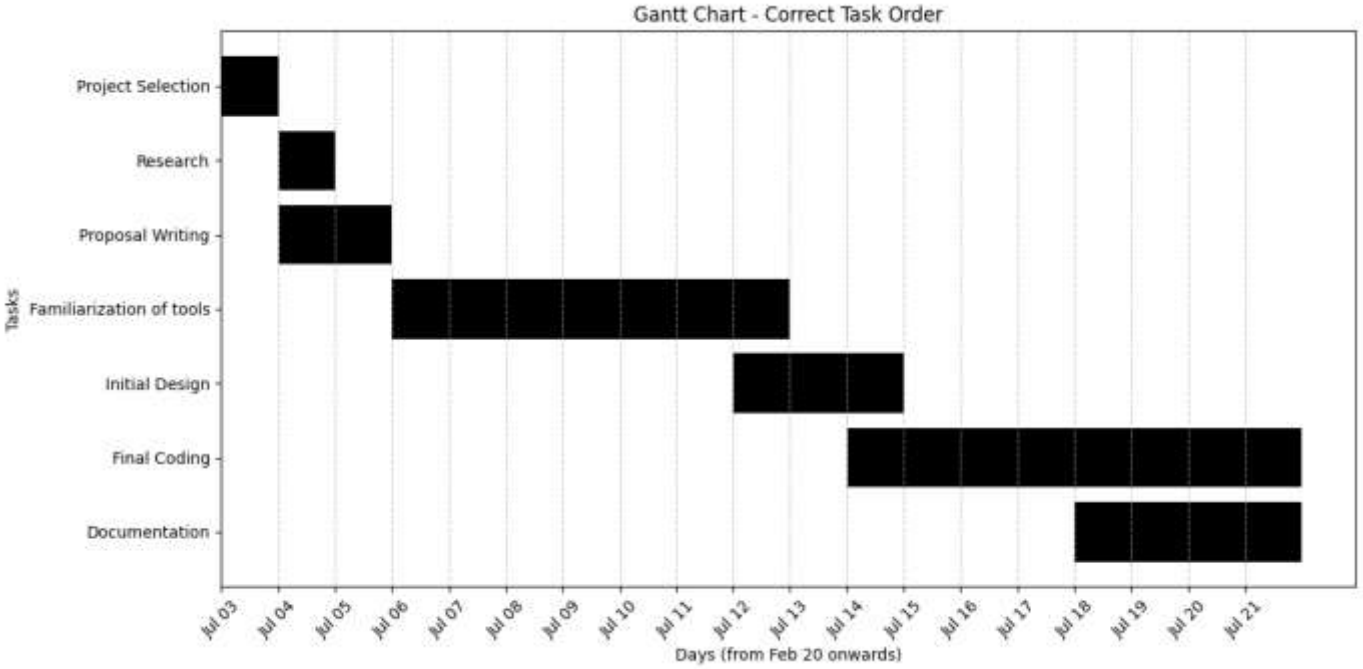


Fig: Gantt Chart

8. REFERENCES

1. *Sudoku Rules and Strategies – https://sudoku.com*
   (Understanding puzzle logic, rules, and variations)
2. *Object-Oriented Programming in C++, 4th Edition – Robert Lafore*
   (For design patterns and OOP implementation in C++)
3. *Qt for Beginners – Official Documentation – https://doc.qt.io/qt-6/gettingstarted.html*
   (GUI development using Qt in C++)
4. *Backtracking Algorithm Explained – GeeksforGeeks*
   https://www.geeksforgeeks.org/sudoku-backtracking/
   (Used for implementing Sudoku solving logic)
5. *Stack Overflow – https://stackoverflow.com*
   (Code troubleshooting and logic help)
6. *Cplusplus.com – https://www.cplusplus.com/*
   (Standard C++ reference and STL documentation)