TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A Project Report 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
Lalitpur, Nepal

August 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 Object Oriented Programming Teacher Daya Sagar 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)

ABSTRACT

This project presents the design and development of a Sudoku game application implemented in C++ using the SFML 3.0.0 graphics library. The primary objective of the project is to combine the concepts of Object-Oriented Programming **(**OOP**)** with practical problem-solving in an interactive environment. The system features a user login and registration mechanism that stores player information, including score, diamonds, and unlocked levels, in a persistent text-based file. Players can choose difficulty levels, play Sudoku under time-bound conditions, and earn or lose diamonds based on their performance. A scoring system is integrated to reward faster completion and efficient gameplay while penalizing diamond usage.

Additional features include a ranking system for comparing user performance, level progression with unlocking new stages, and an intuitive menu-driven interface. The project emphasizes concepts such as file handling, stream formatting, operator overloading, and user-defined functions, while also providing practical exposure to event handling and graphics programming.

The successful completion of this project demonstrates how C++ can be effectively used to create engaging applications that blend logic, data persistence, and interactive graphics.

Table of Content

1. 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 backtracking algorithm capable of automatically solving valid puzzles.
- To demonstrate clean, maintainable, and reusable code architecture suitable for further enhancements or feature additions.

2. 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 Users, Menu, levelSelection, and SudokuGame, the project will demonstrate important OOP concepts such as classes, objects, encapsulation, modularity and abstraction. 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 along with user login interface and level selection methods.

The scope of this project is limited to desktop gameplay with small screen 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.

2.2 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.

2.2 History

Although often linked with Japan, the origins of Sudoku trace back to number puzzles created in France during the late 18th century, where digits were arranged in grids without repetition across rows or columns. The modern version emerged in 1979 when American architect Howard Garns introduced a puzzle called "Number Place" in Dell Puzzle Magazine. A few years later, in 1984, the Japanese publishing company Nikoli adapted the game, giving it the name Sudoku, a shortened form of the phrase "Sūji wa dokushin ni kagiru," meaning "numbers must remain single." Its clear structure, logical reasoning, and wide accessibility quickly made it a cultural phenomenon in Japan. By the early 2000s, Sudoku had spread globally, largely due to its presence in newspapers, magazines, and online puzzle collections. Today, Sudoku continues to be one of the most popular and enduring logic-based games worldwide, valued for its universal appeal, scalability in difficulty, and ability to engage players of all ages**.**

2.3 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,
- A user login system.
- Ranking system of user based on the score they achieved
- Diamond System and Time Limitation.
- An automated solving algorithm (e.g., backtracking),
- Level Based Game Difficulty.
- 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.

3. APPLICATION

- Provides entertainment and interactive puzzle-solving experience.
- Enhances logical thinking and problem-solving skills.
- Acts as an educational tool suitable for learners of all ages.
- Useful for demonstrating backtracking and problem-solving techniques in C++.
- Demonstrates programming concepts such as Object-Oriented Programming, file handling, and graphical interface development using SFML.
- Introduces gamification elements like scores, diamonds, levels, and rankings to motivate users.
- Can serve as a foundation for future extensions like mobile versions, online leaderboards, or AI-assisted Sudoku solvers.

4. LITERATURE SURVEY

Sudoku is a popular logic-based puzzle with numerous digital implementations. Early versions were text-based, while modern games use graphical interfaces for better interactivity. Puzzle generation often employs backtracking or constraint-based algorithms to ensure solvable and challenging boards.

Digital Sudoku games commonly include features like difficulty levels, timers, scoring systems, hints, and user progress tracking. These features enhance engagement and promote logical thinking and problem-solving skills.

The current project builds on these ideas by providing a C++ SFML-based Sudoku game with multiple difficulty levels, timed gameplay, diamond penalties for incorrect moves, and a file-based user login system to save scores and progress, offering an interactive and user-friendly experience.

5. 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.

6. 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:

6.1. Requirement Analysis

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

6.2. System Design

Use modular class-based design to separate responsibilities such as: Level class for managing the selection of level,

- User class for handling user information, storing and retrieving from files,
- Menu and level class for selection ,
- SudokuGame class for implementing puzzle, board generation and backtracking algorithm.

6.3. 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.

6.4. Algorithm Implementation

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

6.5. 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.

7. IMPLEMENTATION

The proposed system was 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.

The designed system was almost similar to proposed system with some more features added. Here are some of the detail explaination:

7.1 Features:

i. Tools and Technologies Used

- Programming Language: C++
- Graphics Library: SFML 3.0
- File handling for user data storage

ii. System Architecture

- Describe the main modules of the game:
    - User Management (login, create new user, save progress)
    - Sudoku Game Logic (board generation, puzzle solving, validation)
    - Menu System (buttons, navigation)
    - UI Rendering (displaying board, numbers, diamonds, timer, score)

iii. Algorithm and Logic

- Sudoku board generation using backtracking algorithm
- Puzzle creation by removing numbers based on difficulty
- Input handling: keyboard and mouse
- Scoring and diamond reduction mechanism
- Timer functionality and game-over conditions

iv. Real-Time Rule Validation

Invalid moves are detected instantly — the system ensures no repetition of digits in rows, columns, or 3x3 sub grids.

v. Class Design and OOP Features

- Classes implemented: User, Menu, SudokuGame
- Explain attributes and methods of each class
- Highlight use of encapsulation, modularity, and function-based separation

vi. File Handling

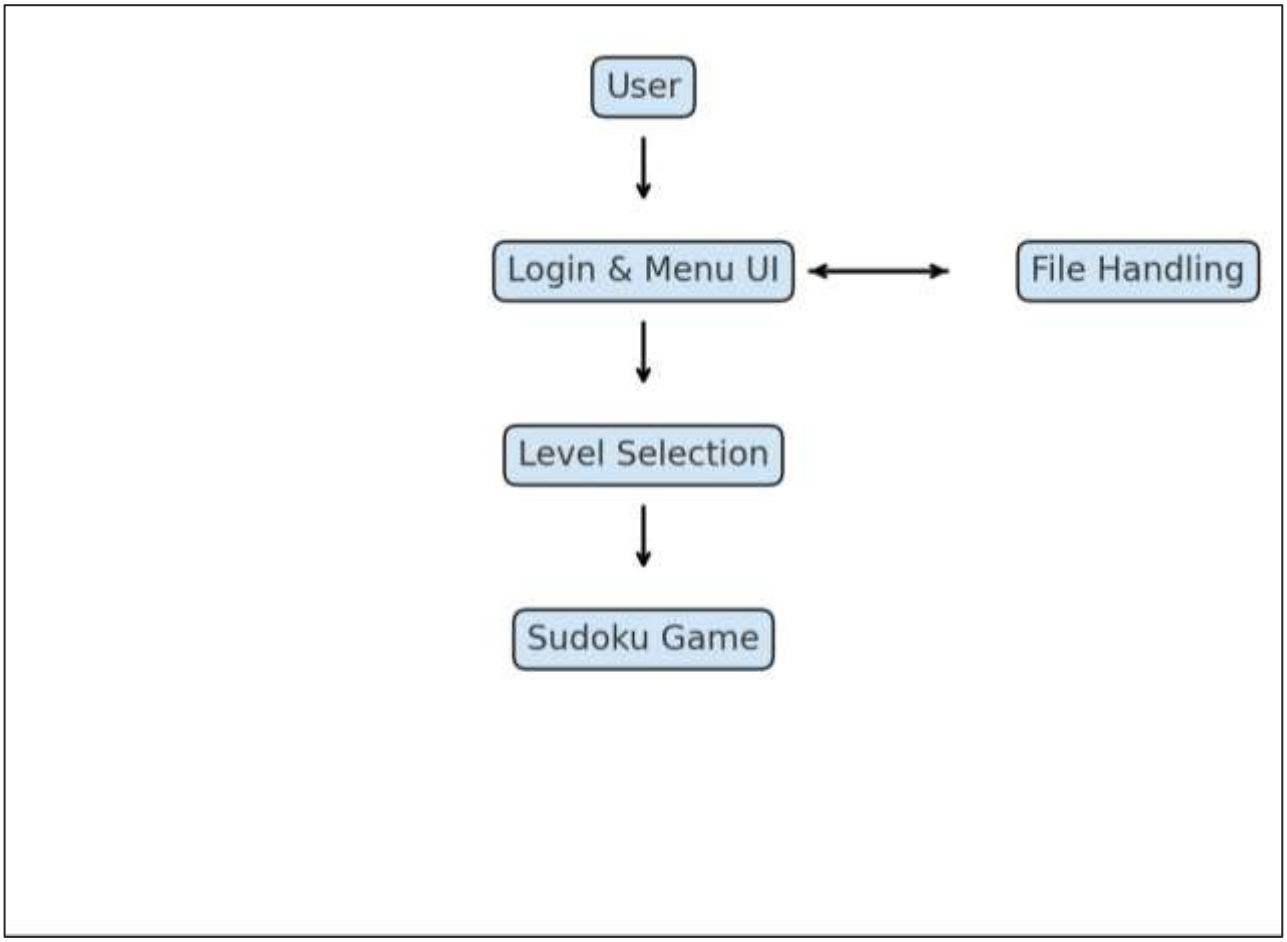
- Storing and reading user data (users.txt)
- Updating user progress: score, unlocked levels, diamonds

vii. User Interface Implementation

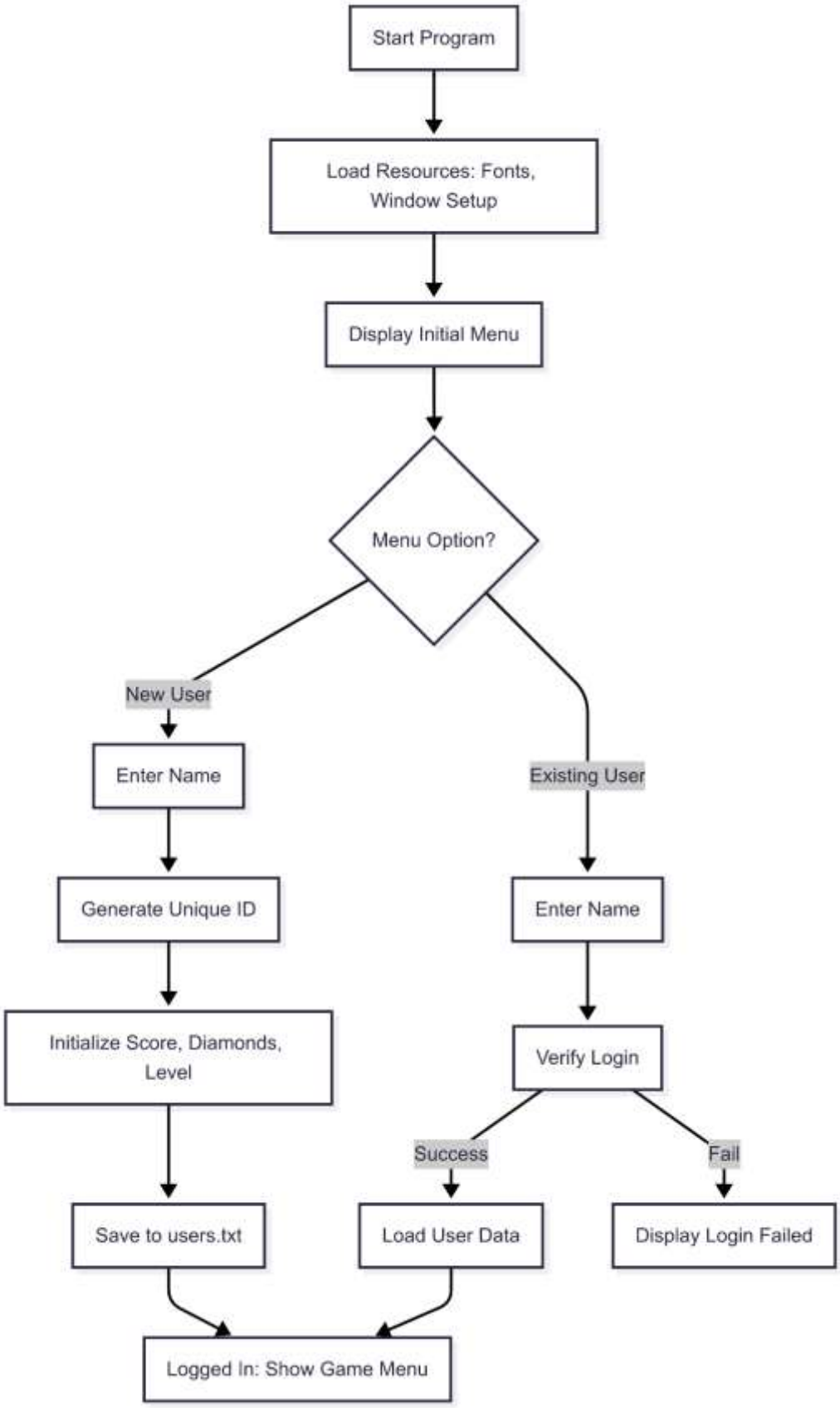
- Layout of Sudoku board and menus
- Displaying score, diamonds, and remaining time
- Interactive feedback for correct/incorrect inputs

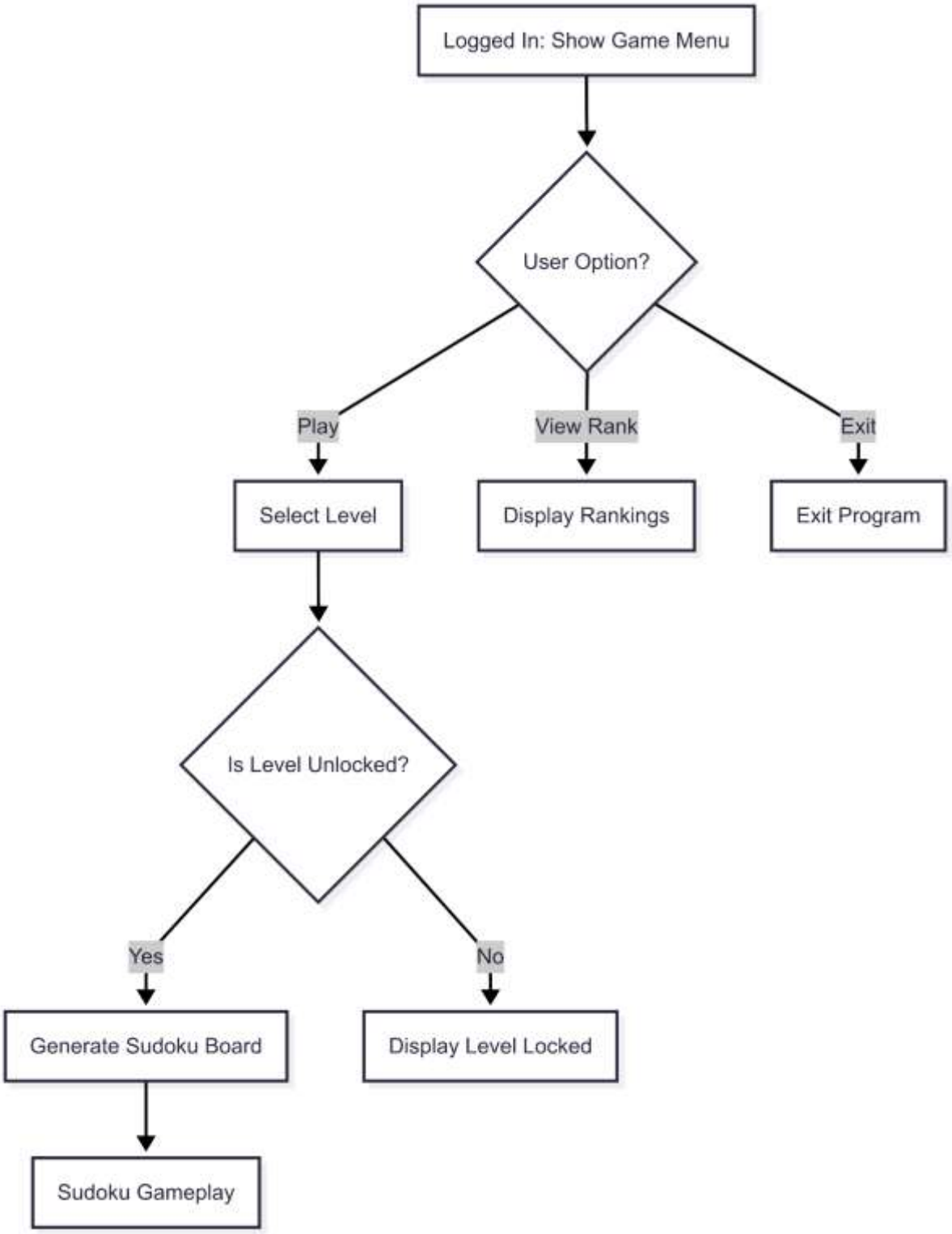
.

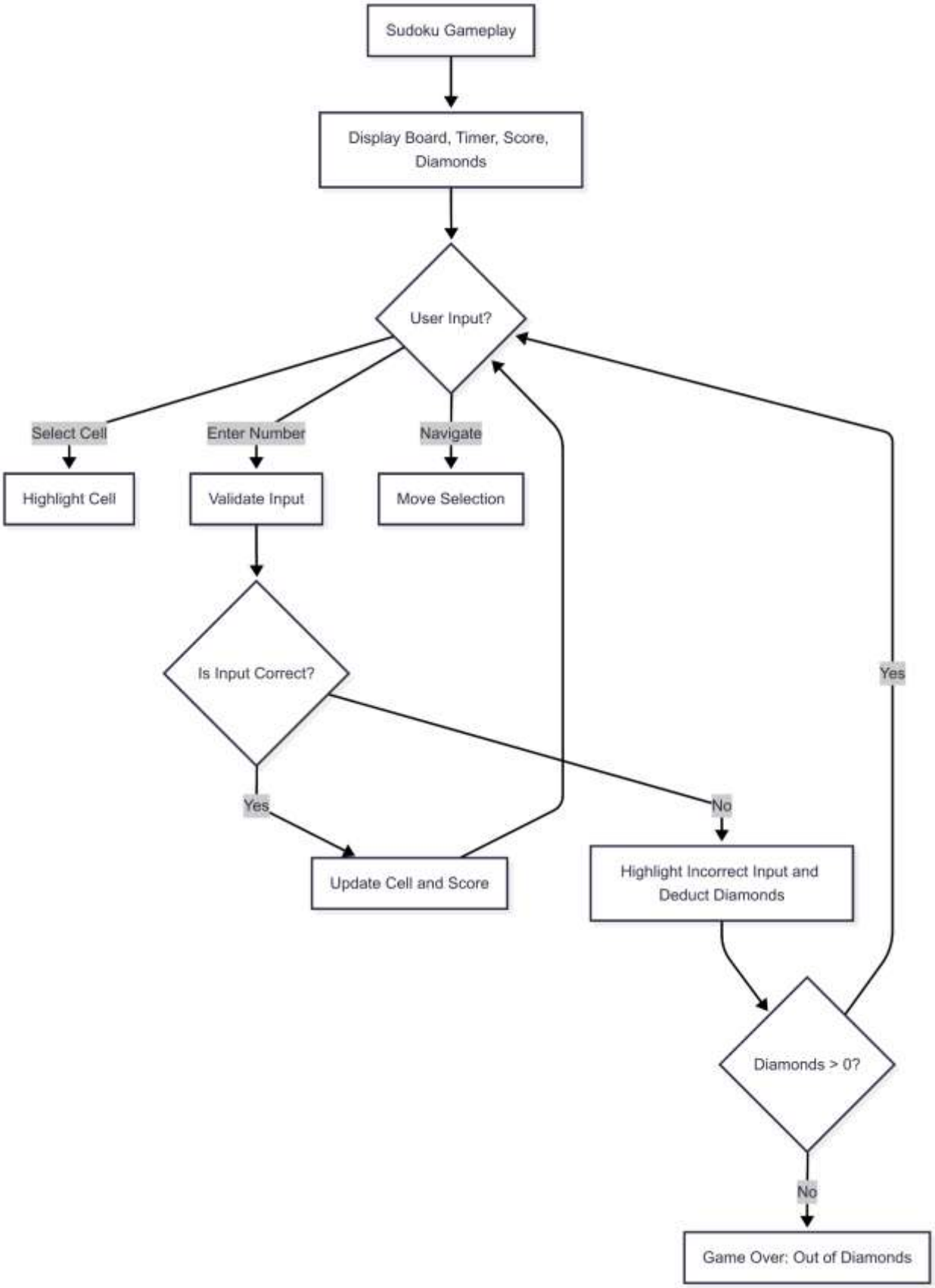
7.2 Block Diagram

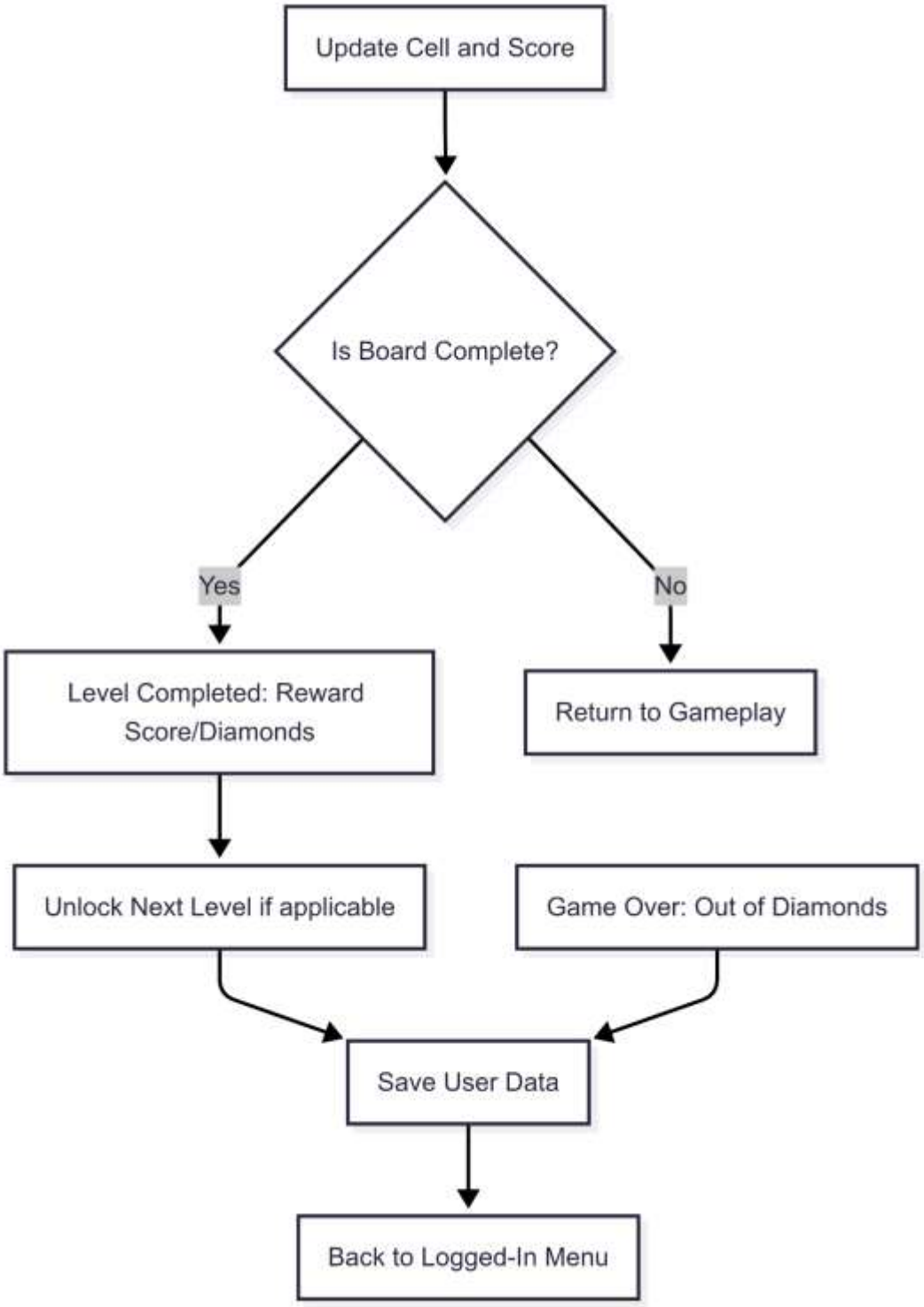


7.3 Flowchart




Logged In: Show Game Menu
User Option?
Play
View Rank
Exit
Select Level
Display Rankings
Exit Program
Is Level Unlocked?
Yes
No
Generate Sudoku Board
Display Level Locked
Sudoku Gameplay


Sudoku Gameplay
Display Board, Timer, Score, Diamonds
User Input?
Select Cell
Enter Number
Navigate
Highlight Cell
Validate Input
Move Selection
Is Input Correct?
Yes
No
Update Cell and Score
Highlight Incorrect Input and Deduct Diamonds
Diamonds > 0?
Yes
No
Game Over: Out of Diamonds


Update Cell and Score
Is Board Complete?
Yes
No
Level Completed: Reward Score/Diamonds
Return to Gameplay
Unlock Next Level if applicable
Game Over: Out of Diamonds
Save User Data
Back to Logged-In Menu

8. RESULT

The following section demonstrates the various stages and functionalities of the Sudoku game through screenshots, highlighting the user interface, gameplay, and progress tracking.

8.1. Login / New User Creation

- Users can either log in with existing credentials or create a new profile.
- The program validates login attempts and initializes new users with default score, diamonds, and unlocked level.

8.2. Main Menu

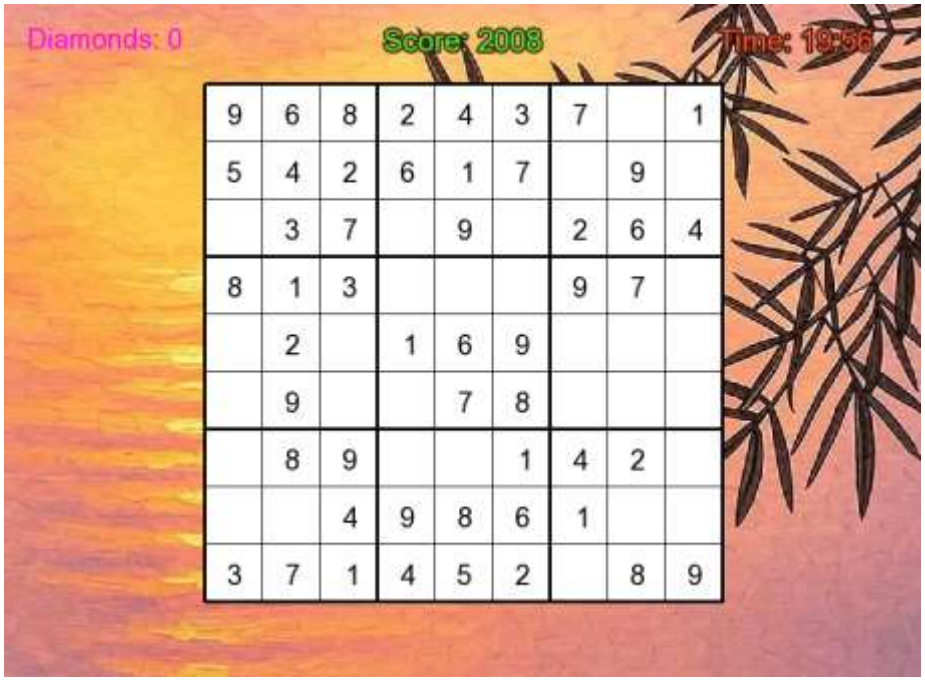
- Once logged in, users can choose to play, view rankings, or exit.
- The interface clearly displays available options.

8.3. Level Selection

- Users select a level to play. Locked levels are indicated, and only unlocked levels can be accessed.
- This ensures structured progression.

8.4. Sudoku Gameplay

- The game board is displayed with empty cells for the player to fill.
- Current score, remaining diamonds, and timer are displayed at the top.
- Users interact using mouse clicks and keyboard inputs.



8.5. Input Validation

- Correct number entries update the board and increase score.
- Incorrect entries are highlighted in red, and diamonds are reduced.
- If diamonds reach zero, the game ends.

8.6. Level Completion and Rewards

- Successfully completing a level rewards the user with score points and additional diamonds.
- The next level is unlocked automatically if applicable.

8.7. Game Over / Time Up

- If time expires or diamonds reach zero, the game ends and displays the game-over message.
- User progress is saved automatically.

8.8. Ranking Display

- Users can view rankings to compare scores.

This is all the results we obtained in our game Sudoku.

9. PROBLEM FACED AND SOLUTION

While developing the Sudoku game, we encountered several challenges. The SFML 3.0.0 library was new to us, and there were limited tutorials to guide learning its graphics and input handling. Implementing the backtracking algorithm for Sudoku generation was also unfamiliar and required careful debugging. Managing dynamic user data like scores, diamonds, and unlocked levels while ensuring persistence across sessions proved tricky. Additionally, integrating real-time gameplay features such as timers, cell selection, and input validation required a solid understanding of event handling. Combining all these elements into a smooth, functional game tested both our programming and problem-solving skills.

To overcome these challenges, we studied SFML 3.0.0 through its official documentation and experimented with its graphics and input functions. The backtracking algorithm for generating Sudoku boards was implemented step by step, ensuring correctness through testing. We designed a file-based user management system to save and load scores, diamonds, and unlocked levels efficiently. Real-time gameplay mechanics, including timers, cell selection, and input validation, were integrated using SFML's event handling. By modularizing the code into classes like User and SudokuGame, we maintained clean, manageable code. This systematic approach allowed us to successfully develop a fully functional, interactive Sudoku game.

10. LIMITATION AND ENHANCEMENT

Limitations:

The Sudoku game is limited to a standard 9x9 grid, which restricts the variety of puzzles that can be played. The difficulty levels are predefined, and they may not perfectly cater to all skill levels of players. The user interface, built using SFML 3.0.0, is basic and lacks advanced visual effects or animations. The game only supports single-player mode and does not include multiplayer or online features. The scoring and diamond system depends entirely on correct input and does not reward partial progress or hints. Additionally, the game stores user data locally in a users.txt file, which may not be secure or suitable for large-scale use. There is also no undo feature, meaning mistakes must be corrected manually.

Enhancements:

Future enhancements could include support for different grid sizes and Sudoku variants, making the game more versatile. Adding adjustable or adaptive difficulty levels could better cater to players of varying skill levels. The user interface could be improved with animations, effects, and sound to make the gameplay more engaging. Implementing multiplayer or online leaderboards would add a competitive element. A more robust scoring system that considers hints or partial

solutions could be introduced. User data storage could be shifted to a database for improved security and scalability. Finally, features like undo, redo, and hints could be added to enhance user experience and gameplay flexibility.

11. CONCLUSION AND RECOMMENDATIONS

Conclusion:

The Sudoku game project successfully demonstrates the implementation of a digital logic puzzle with features such as multiple difficulty levels, score tracking, diamond-based penalty system, and user management. Through this project, key concepts of C++ programming, file handling, object-oriented programming, and graphical interface design using SFML 3.0.0 were applied effectively. The game provides an engaging and educational experience for users, allowing them to improve logical thinking and problem-solving skills.

Recommendations:

For future improvements, it is recommended to enhance the user interface with animations and sound effects to make the game more visually appealing. Additional features such as multiple grid sizes, different Sudoku variants, undo/redo functionality, and online leaderboards can increase user engagement. Migrating user data storage to a database will improve scalability and security. Implementing adaptive difficulty and hint systems could make the game more accessible to a wider audience, providing both beginners and advanced players with a more enjoyable experience.

12. 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. *SFML 3.0 Tutorials – https://www.sfml-dev.org/tutorials/3.0/*
   (For graphics, window management, and SFML-based game development)

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)