ACKNOWLEDGEMENT

The successful completion of this project would not have been possible without the guidance, support, and assistance of many individuals. We are deeply grateful to all who contributed their time, knowledge, and encouragement throughout this journey.

First and foremost, we would like to express our sincere gratitude to our lecturer, **Mr. Jalauddin Mansur Sir**, for his invaluable guidance, unwavering support, and continuous encouragement, which played a vital role in accomplishing this project.

We also extend our heartfelt thanks to our classmates and friends who showed great interest in our work, offered valuable suggestions, and provided essential information that greatly contributed to the quality and success of our project.

Our sincere appreciation goes to our senior brothers and sisters for their constant encouragement, timely advice, and support, which motivated us to complete this work successfully.

Lastly, we would like to express our sincere thanks to the **Department of Electronics and Computer Engineering** for providing us with the platform and resources to undertake and complete this project.

ABSTRACT

This project presents the development of a **KeyMaster: A Graphical Typing Challenge**, designed to evaluate and improve a user's typing speed and accuracy in an engaging and interactive manner. The game challenges players to type randomly generated words within a given time limit, measuring both typing speed (words per minute) and accuracy in real time. Implemented in C++ with a graphical user interface, the game features a visually appealing layout, background music, and real-time score updates to enhance user engagement.

To further motivate players, a **leaderboard system** has been integrated, which stores and displays the top five high scores along with player names. Words for the game are loaded from an external dictionary file, ensuring variety and replayability. The system also includes a timer mechanism, error tracking, and immediate feedback to help players identify areas for improvement.

This project demonstrates the practical application of programming concepts such as file handling, data structures, event-driven programming, and GUI development, while providing an enjoyable platform for users to enhance their typing skills.

Table of Contents

1. INTRODUCTION

Typing has become an essential skill in today's digital era, where efficient and accurate text input plays a vital role in professional, academic, and personal communication. The ability to type quickly and precisely not only saves time but also improves productivity. However, developing strong typing skills requires consistent practice and an engaging learning environment.

The **KeyMaster: A Graphical Typing Challenge** is designed as an interactive platform that allows users to test, track, and enhance their typing speed and accuracy in a fun and competitive way. Unlike traditional typing practice tools, this game introduces a time-bound challenge where players are required to type a sequence of randomly selected words within a given time limit. The system then evaluates their performance based on **words per minute (WPM)** and **accuracy percentage**.

Developed in C++ with a graphical interface, the game features an attractive design, background music, and instant feedback mechanisms to keep users motivated. It incorporates a **leaderboard system** that records and displays the top five high scores with player names, encouraging healthy competition and repeated gameplay. The use of an external dictionary file ensures that the word list is varied and replayable, making the game suitable for continuous skill improvement.

This project not only serves as an enjoyable typing challenge but also showcases the practical application of programming concepts such as file handling, data structures, event-driven programming, and GUI development. By combining entertainment with education, the **KeyMaster: A Graphical Typing Challenge** offers a productive and engaging way to improve typing proficiency.

2. OBJECTIVES

The main objectives of the **KeyMaster: A Graphical Typing Challenge** project are as follows:

1. **To create an interactive platform** that allows users to test and improve their typing speed and accuracy in an engaging manner.
2. **To implement real-time performance evaluation** by calculating Words Per Minute (WPM) and accuracy percentage during gameplay.
3. **To enhance user engagement** through a graphical interface, background music, and instant feedback mechanisms.

4. **To develop a leaderboard system** that records and displays the top five high scores along with player names, promoting healthy competition.
5. **To utilize an external dictionary file** for loading random words, ensuring variety and replayability.
6. **To apply core programming concepts** such as file handling, data structures, timers, and event-driven programming in C++.
7. **To encourage skill development** by providing an enjoyable and productive platform for improving typing proficiency.

3. SCOPE

The **KeyMaster: A Graphical Typing Challenge** project is designed to serve both educational and entertainment purposes, targeting users who wish to enhance their typing speed and accuracy. It can be used by students, professionals, and individuals preparing for typing-based tests or improving their general keyboard skills. The game is built with scalability in mind, allowing for future enhancements such as different difficulty levels, multiplayer modes, and online scoreboards.

Key features within the current scope include:

- Real-time measurement of **Words Per Minute (WPM)** and **accuracy**.
- Randomized word generation from an external dictionary file.
- A **leaderboard** storing the top five high scores with player names.
- Graphical user interface (GUI) with visual feedback and background music.
- Compatibility with standard computer systems running the game in C++.

4. METHODOLOGY

The development of the **KeyMaster: A Graphical Typing Challenge** followed a structured approach to ensure functionality, usability, and performance. The major steps include:

1. **Requirement Analysis**
    - Identified key features such as WPM calculation, accuracy tracking, timer, leaderboard, and GUI integration.
    - Selected C++ as the programming language with a graphics library for the interface.
2. **Design Phase**
    - Created the game flowchart and interface layout.
    - Designed the data structure for storing dictionary words, player names, and scores.
3. **Implementation**
    - Developed core game logic for random word generation, real-time typing input, and scoring.
    - Integrated file handling to read words from an external dictionary and store leaderboard data.
    - Added graphical components, timer display, background music, and feedback messages.
4. **Testing & Debugging**
    - Conducted multiple gameplay tests to ensure accurate WPM and error calculation.
    - Fixed bugs related to timing, score updates, and file operations.
5. **Finalization**
    - Optimized the game for smooth performance.
    - Prepared documentation including the abstract, introduction, objectives, and technical details.

5. IMPLEMENTAION

The implementation of the **KeyMaster: A Graphical Typing Challenge** follows a structured approach to ensure efficiency, responsiveness, and user engagement. The development process includes designing an interactive graphical interface, implementing real-time typing speed and accuracy calculations, and maintaining a leaderboard for high scores. The key phases of development include:

1. Word Management:

- Words are loaded from an external text file (words.txt) into a vector using file handling in C++.
- The word list is shuffled to ensure random selection for each gameplay session.
- A pointer keeps track of the current word, which is updated once the user completes typing the previous word.

2. User Interaction:

- Users type the displayed word in the input area using the keyboard.
- The game compares typed characters with the target word in real time, highlighting errors.

3. Scoring and Accuracy Calculation:

- Accuracy is computed as:

$$Accuracy(\%) = \frac{\text{Correct Characters}}{\text{Total Typed Characters}} \times 100$$

- The score is updated dynamically as the user types.

4. Leaderboard System:

- After each game, the user is prompted to enter their name.
- The score (WPM and accuracy) is stored in a leaderboard file (leaderboard.csv).
- The leaderboard maintains the **top five scores**, sorted primarily by WPM and secondarily by accuracy.
- The leaderboard is displayed on the game over screen.

5. Rendering:

- The graphical user interface is created using graphics.h, displaying the target word, user input, timer, and live statistics.
- Correctly typed characters are shown in a standard color, while incorrect characters are highlighted in red.
- A progress bar represents the remaining time visually.
- Background music and sound effects (using SFML) enhance the gaming experience.

6. Code Explanation:

a. Initialization and Setup:

- initwindow() initializes the graphics window.
- Dictionary words are loaded using loadWordsFromFile().
- The leaderboard is loaded from file at the start of the program.
- Background music is played using SFML Audio.

b. Handling User Input:

- handleKeyPress(char key): Records typed characters and updates the input buffer.
- Backspace removes the last character from the typed input.
- Writing the complete submits the current word for evaluation.

c. Word Evaluation:

- submitWord(): Compares typed input with the target word, updates correct/incorrect counts, and fetches the next word.
- Plays "correct" sound if the word matches, otherwise plays an "error" sound.

d. Leaderboard Management:

- pushScore(name, wpm, accuracy): Inserts a new score and sorts the leaderboard.
- saveLeaderboard(): Writes updated scores to leaderboard.csv.
- displayLeaderboard(): Renders the top five scores on the screen.

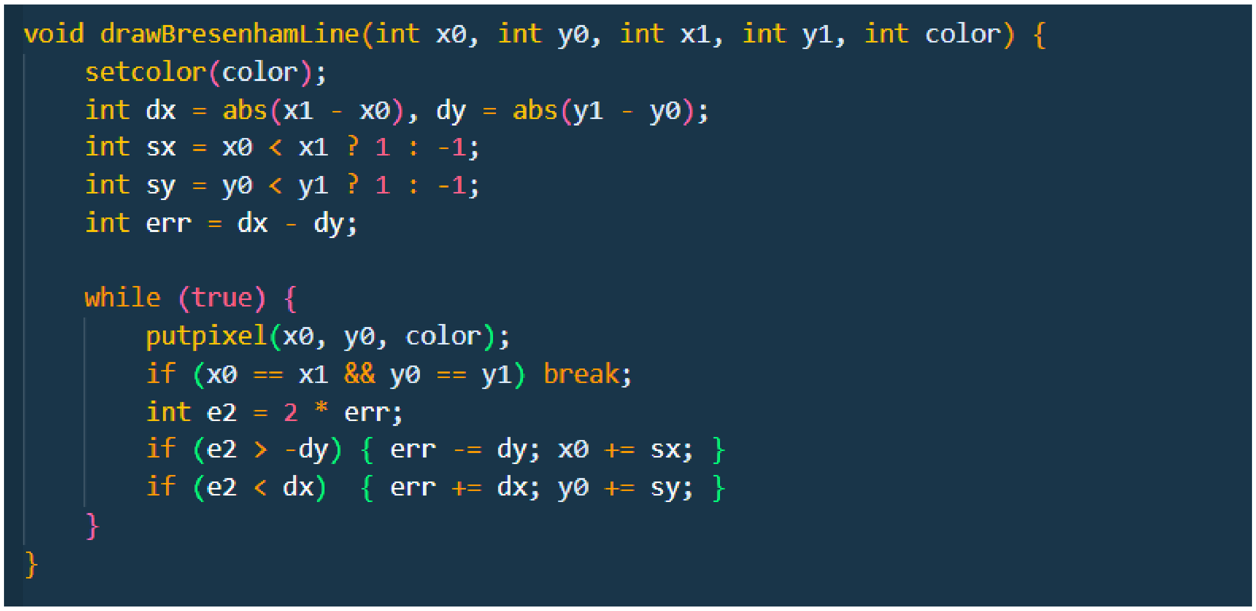
e. Rendering Text and Interface:

- drawWordPanel(): Displays the target word and highlights typed characters.

- drawStats(): Shows WPM, accuracy, and errors in real time.
- drawTimer(): Displays remaining time visually and numerically.

7. Algorithms used

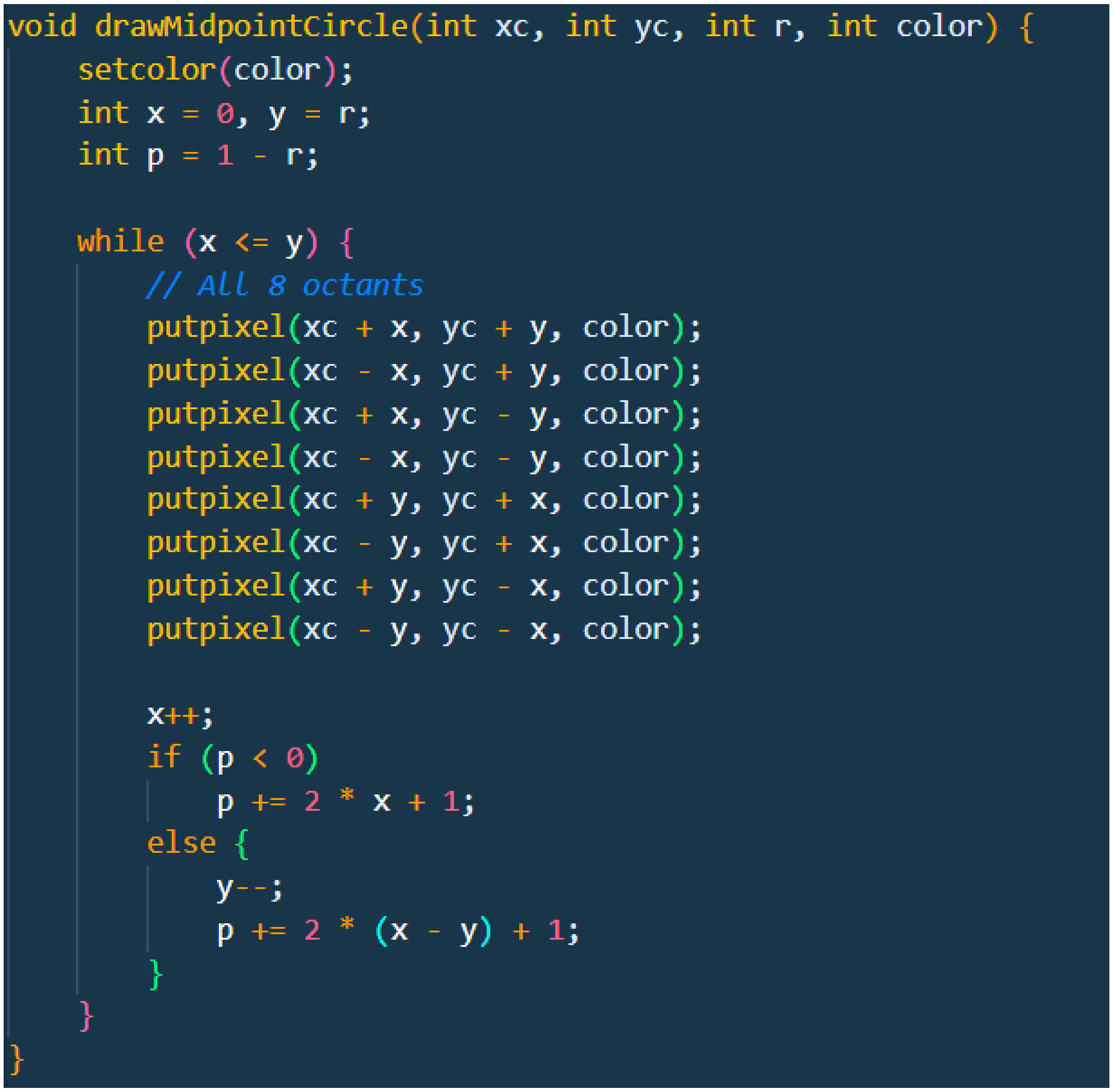
7.1 Bresenham Line drawing algorithm



```
void drawBresenhamLine(int x0, int y0, int x1, int y1, int color) {
    setcolor(color);
    int dx = abs(x1 - x0), dy = abs(y1 - y0);
    int sx = x0 < x1 ? 1 : -1;
    int sy = y0 < y1 ? 1 : -1;
    int err = dx - dy;

    while (true) {
        putpixel(x0, y0, color);
        if (x0 == x1 && y0 == y1) break;
        int e2 = 2 * err;
        if (e2 > -dy) { err -= dy; x0 += sx; }
        if (e2 < dx)  { err += dx; y0 += sy; }
    }
}
```

7.2 MidPoint Circle drawing algorithm

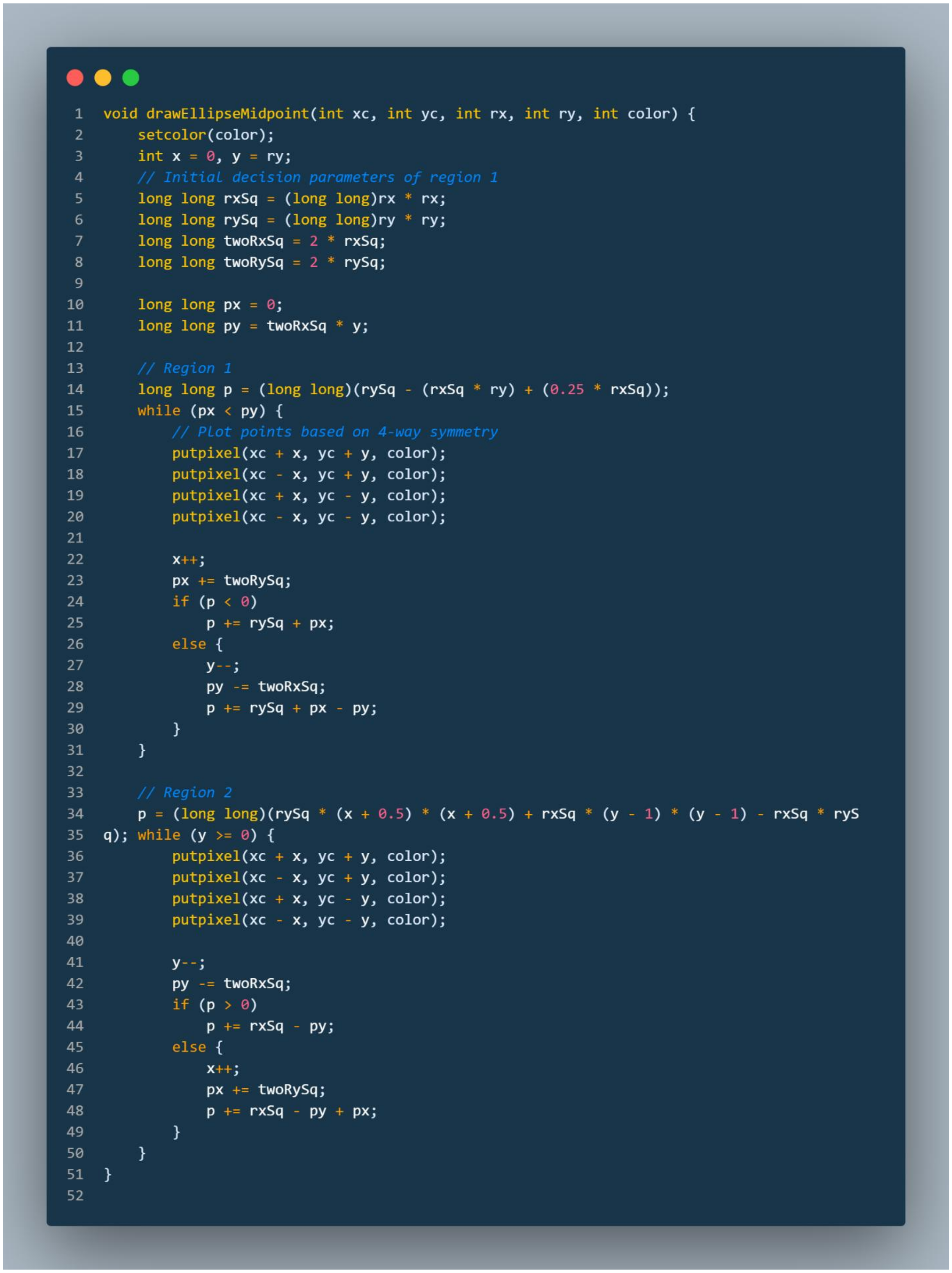


```
void drawMidpointCircle(int xc, int yc, int r, int color) {
    setcolor(color);
    int x = 0, y = r;
    int p = 1 - r;

    while (x <= y) {
        // All 8 octants
        putpixel(xc + x, yc + y, color);
        putpixel(xc - x, yc + y, color);
        putpixel(xc + x, yc - y, color);
        putpixel(xc - x, yc - y, color);
        putpixel(xc + y, yc + x, color);
        putpixel(xc - y, yc + x, color);
        putpixel(xc + y, yc - x, color);
        putpixel(xc - y, yc - x, color);

        x++;
        if (p < 0)
            p += 2 * x + 1;
        else {
            y--;
            p += 2 * (x - y) + 1;
        }
    }
}
```

7.3 MidPoint Ellipse drawing algorithm



```
void drawEllipseMidpoint(int xc, int yc, int rx, int ry, int color) {
    setcolor(color);
    int x = 0, y = ry;
    // Initial decision parameters of region 1
    long long rxSq = (long long)rx * rx;
    long long rySq = (long long)ry * ry;
    long long twoRxSq = 2 * rxSq;
    long long twoRySq = 2 * rySq;

    long long px = 0;
    long long py = twoRxSq * y;

    // Region 1
    long long p = (long long)(rySq - (rxSq * ry) + (0.25 * rxSq));
    while (px < py) {
        // Plot points based on 4-way symmetry
        putpixel(xc + x, yc + y, color);
        putpixel(xc - x, yc + y, color);
        putpixel(xc + x, yc - y, color);
        putpixel(xc - x, yc - y, color);

        x++;
        px += twoRySq;
        if (p < 0)
            p += rySq + px;
        else {
            y--;
            py -= twoRxSq;
            p += rySq + px - py;
        }
    }

    // Region 2
    p = (long long)(rySq * (x + 0.5) * (x + 0.5) + rxSq * (y - 1) * (y - 1) - rxSq * rySq); while (y >= 0) {
        putpixel(xc + x, yc + y, color);
        putpixel(xc - x, yc + y, color);
        putpixel(xc + x, yc - y, color);
        putpixel(xc - x, yc - y, color);

        y--;
        py -= twoRxSq;
        if (p > 0)
            p += rxSq - py;
        else {
            x++;
            px += twoRySq;
            p += rxSq - py + px;
        }
    }
}
```

6. SYSTEM DESIGN

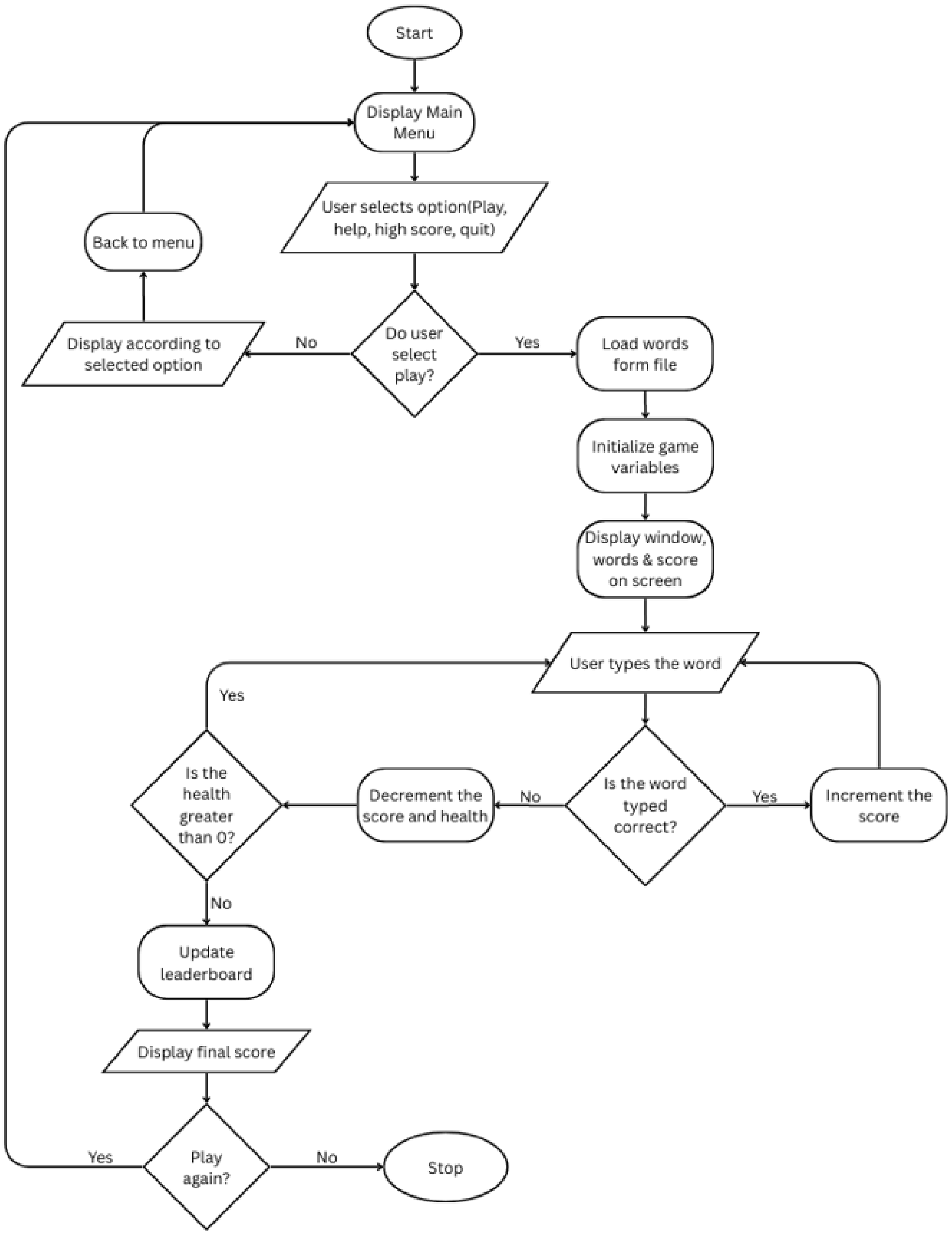
The **KeyMaster: A Graphical Typing Challenge** system is designed to provide a smooth flow from starting the game to displaying results and updating the leaderboard. The design follows a **modular approach**, where each module is responsible for a specific task such as word management, scoring, timing, rendering, and file handling.

6.1 System Components

1. **Input Module**
   - Captures user keystrokes in real time.
   - Handles special keys such as Backspace, Space, and Enter.
2. **Word Management Module**
   - Loads words from the dictionary file.
   - Randomly selects and displays the next target word.
3. **Scoring Module**
   - Calculates Words Per Minute (WPM).
   - Calculates accuracy percentage and error count.
4. **Timer Module**
   - Counts down from the set game duration (e.g., 60 seconds).
   - Triggers game over when time runs out.
5. **Leaderboard Module**
   - Stores and updates top five scores in a file.
   - Displays leaderboard on the game over screen.
6. **Rendering Module**
   - Draws the interface (target word, typed word, statistics, timer, leaderboard).
   - Highlights correct and incorrect characters in different colors.

6.2 Flowchart

Below is the logical flow of the KeyMaster: A Graphical Typing Challenge:



Start
Display Main Menu
User selects option(Play, help, high score, quit)
Back to menu
Do user select play?
No
Yes
Display according to selected option
Load words form file
Initialize game variables
Display window, words & score on screen
User types the word
Yes
Is the health greater than 0?
Decrement the score and health
No
Is the word typed correct?
Yes
Increment the score
No
Update leaderboard
Display final score
Yes
Play again?
No
Stop

7. RESULTS AND DISCUSSION

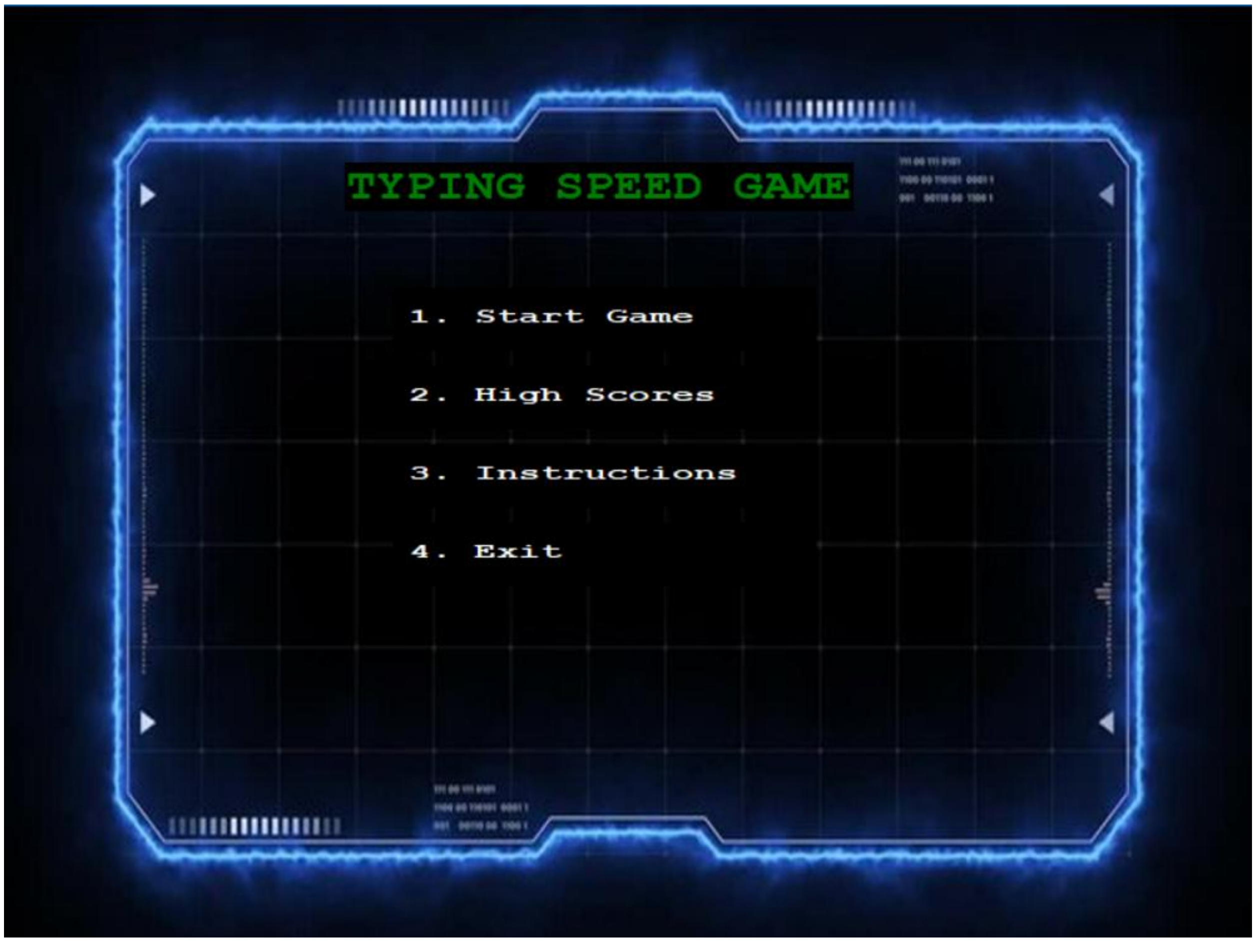
Results:

The KeyMaster: A Graphical Typing Challenge was tested with multiple users to evaluate its performance, accuracy, and usability. The key results observed are as follows:

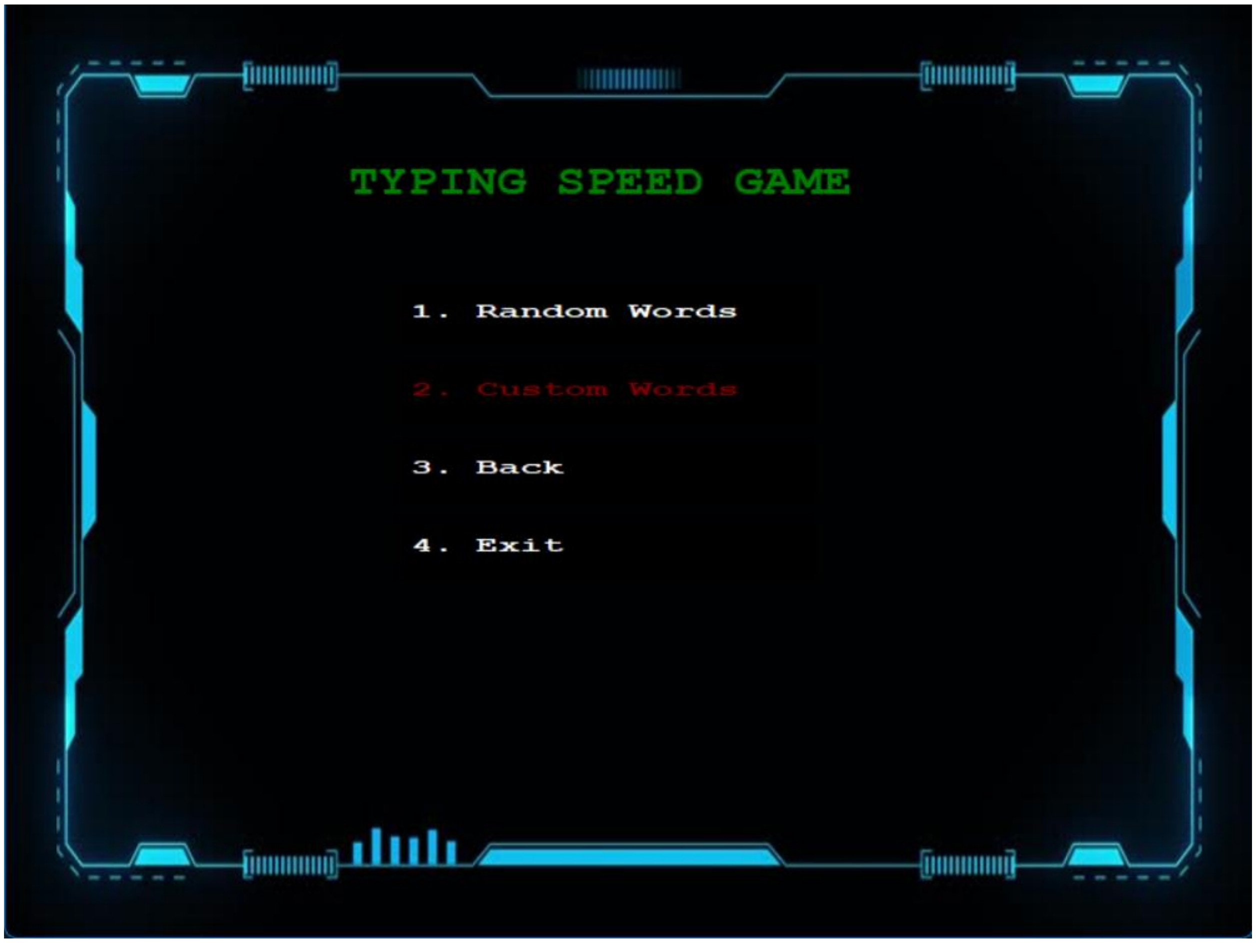
1. **Typing Speed:**
    - Average typing speed among users ranged from **35 to 65 words per minute (WPM)**.
    - Users with prior typing experience achieved higher speeds, while beginners had slower rates.
2. **Accuracy:**
    - The accuracy of typed words was recorded for each user. Most users maintained an accuracy of **85–95%**, with errors mainly occurring in longer or more complex words.
    - The error handling system successfully highlighted incorrect words, allowing users to identify mistakes and improve performance.
3. **High Score Leaderboard:**
    - The top 5 high scores were dynamically updated in real-time. Users could input their names and view rankings, enhancing competitiveness and replay value.
4. **User Interaction and Experience:**
    - Users found the game interface intuitive and responsive. The immediate visual feedback for correct and incorrect typing contributed to an engaging experience.
    - The absence of a timer allowed users to focus on accuracy rather than speed pressure, promoting a more relaxed learning environment.

Outputs:

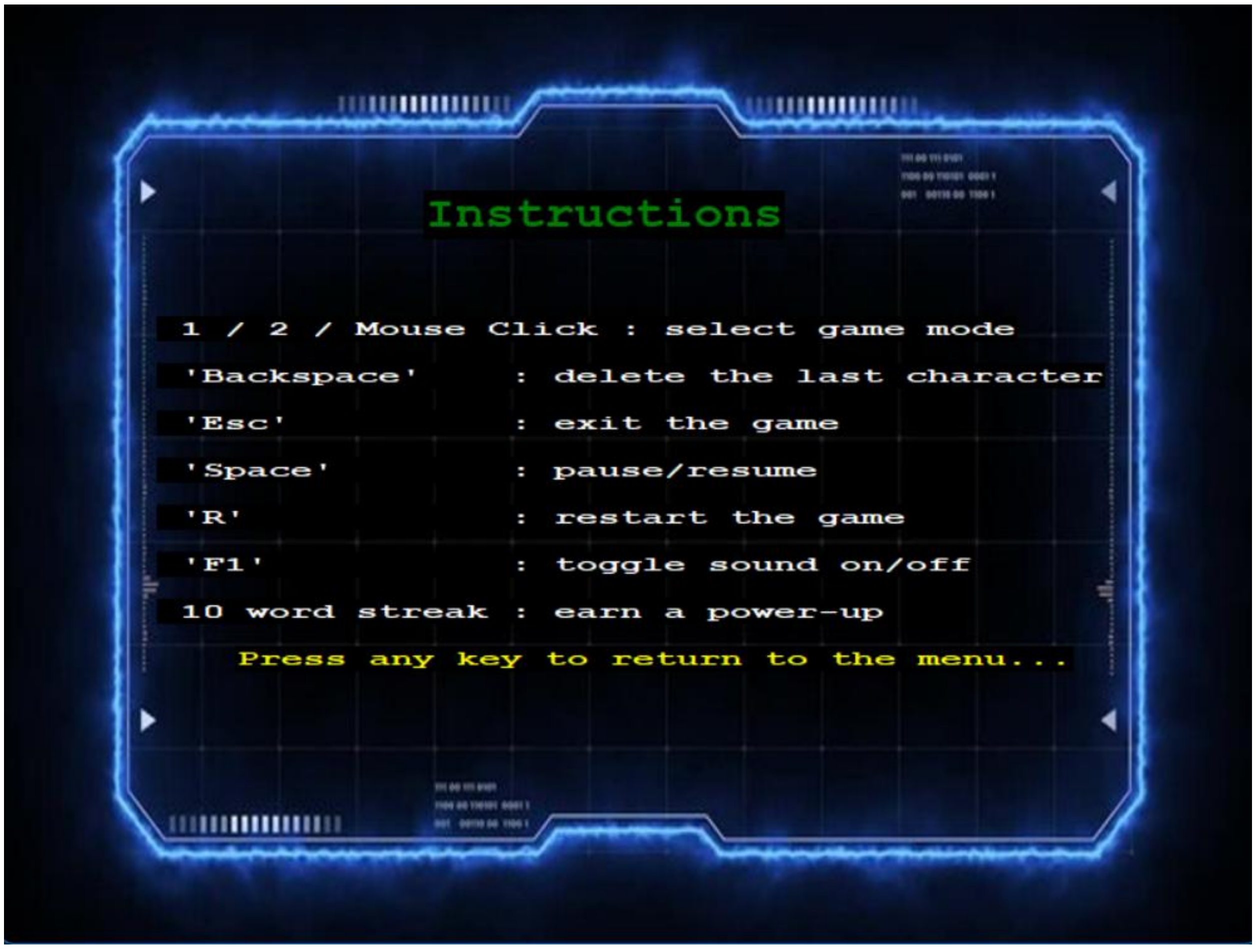
1. **Menu**



2. **Start**



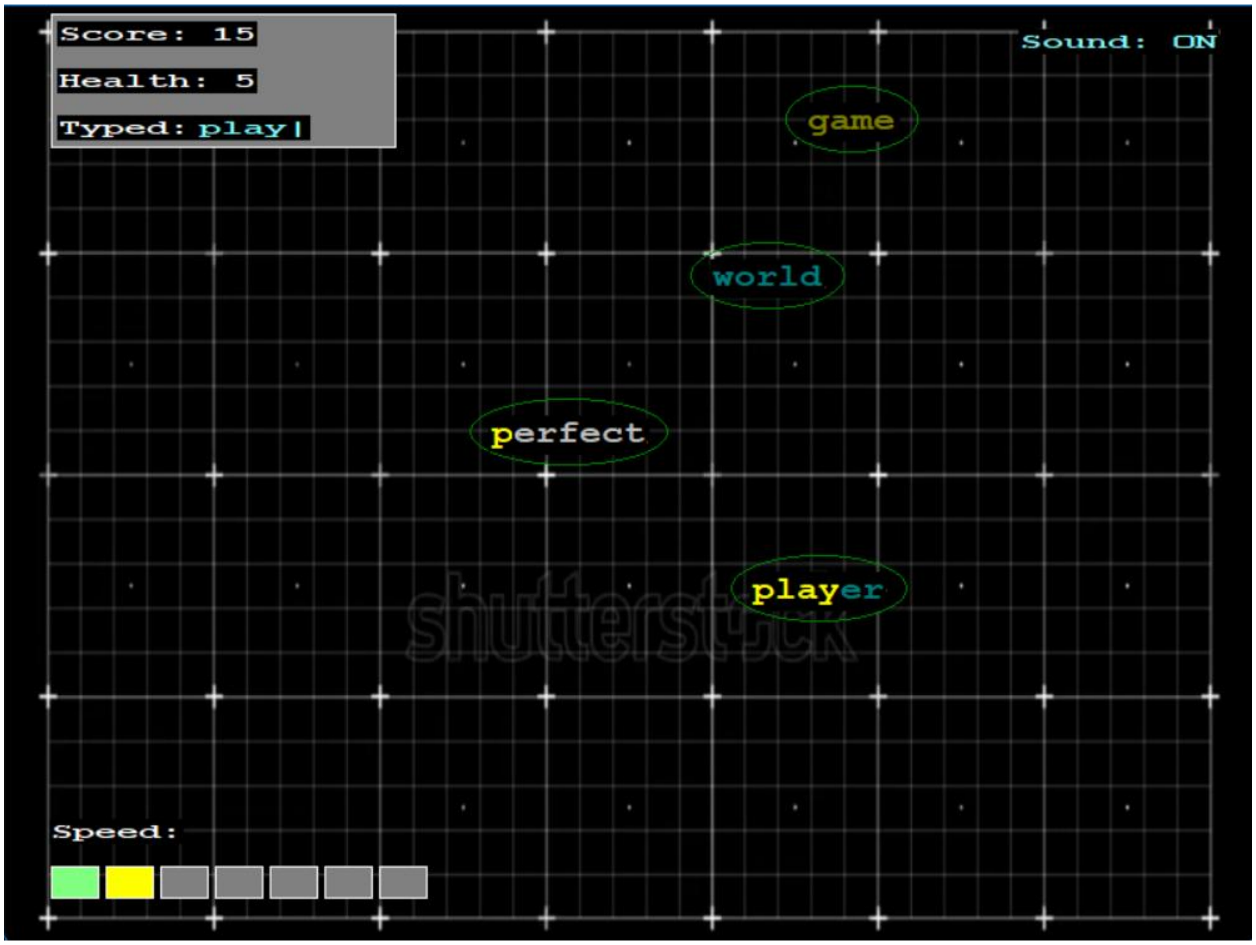
3. **Instruction**



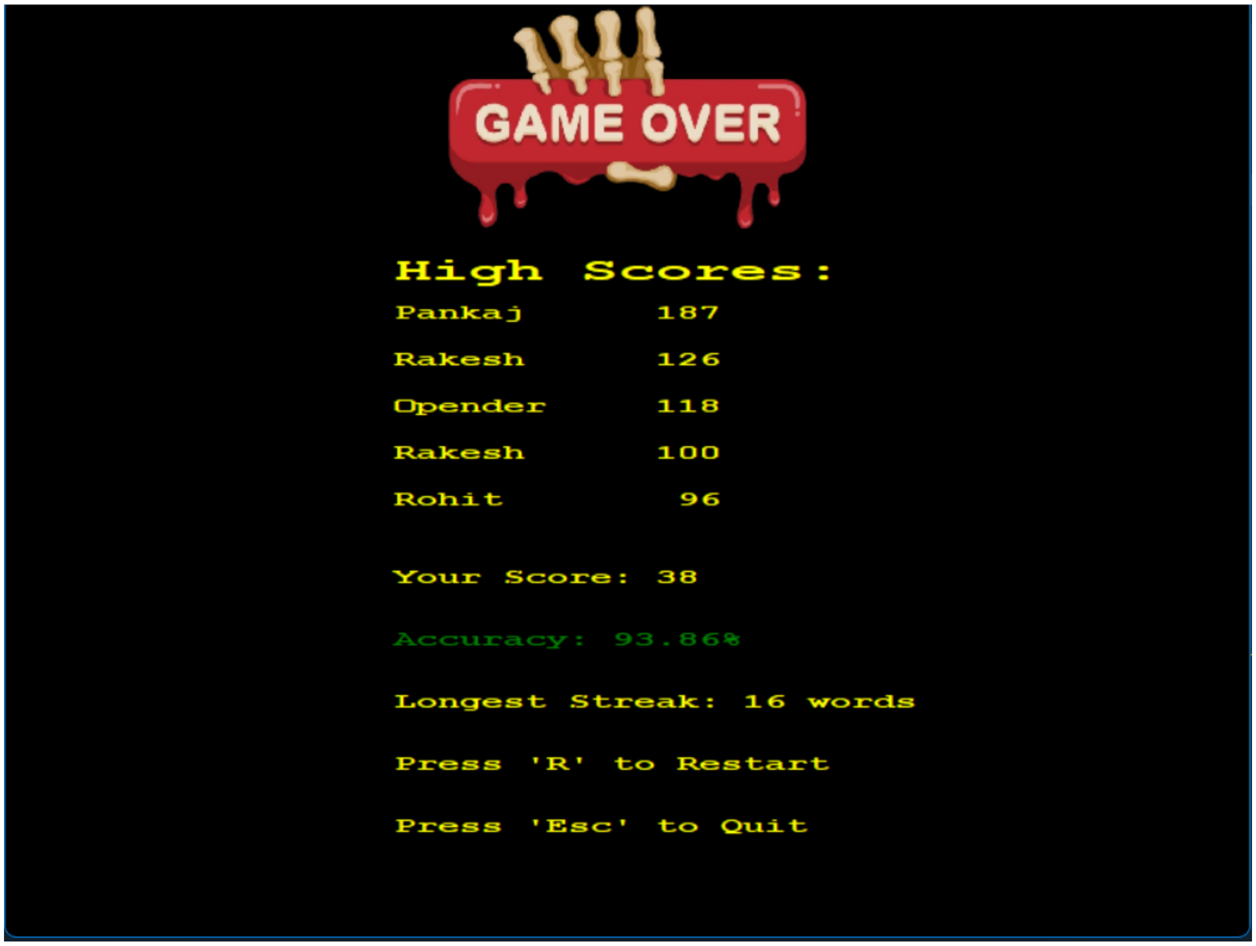
Instructions

```
1 / 2 / Mouse Click : select game mode
'Backspace'         : delete the last character
'Esc'               : exit the game
'Space'             : pause/resume
'R'                 : restart the game
'F1'                : toggle sound on/off
10 word streak      : earn a power-up
    Press any key to return to the menu...
```

4. **In game**


Score: 15
Health: 5
Typed: play|
Sound: ON
game
world
perfect
player
Speed:

5. **Game Over**



```
High Scores:
Pankaj        187
Rakesh        126
Opender       118
Rakesh        100
Rohit          96

Your Score: 38

Accuracy: 93.86%

Longest Streak: 16 words

Press 'R' to Restart

Press 'Esc' to Quit
```

Discussion:

The KeyMaster: A Graphical Typing Challenge effectively combines entertainment with skill development. The results indicate that regular practice using the game can improve both typing speed and accuracy. The inclusion of a leaderboard motivates users to continually improve their performance.

Additionally, the game's design demonstrates the importance of user-friendly interfaces in educational software. Highlighting errors in real-time and providing immediate feedback enhances learning efficiency. While the game does not currently incorporate timed challenges, adding this feature in the future could cater to users aiming to improve their speed under pressure.

Overall, the testing validates that the game is both functional and educational. Minor errors reported by users were mostly due to typing unfamiliar words rather than software faults, indicating robustness in the program's design.

Future Enhancements

The KeyMaster: A Graphical Typing Challenge can be further improved by implementing the following enhancements:

1. **Timed Challenges:**
    - Introduce countdown timers for each round to test users' typing speed under pressure, adding an extra layer of challenge.
2. **Difficulty Levels:**
    - Implement multiple difficulty levels based on word complexity, sentence length, or typing speed targets, catering to both beginners and advanced users.
3. **Multiplayer Mode:**
    - Add a competitive multiplayer feature where users can compete in real-time, enhancing engagement and motivation.
4. **Custom Word Lists:**
    - Allow users to create or upload custom word lists for personalized practice, especially useful for learning specific terminology or languages.
5. **Performance Analytics:**
    - Track detailed statistics such as typing speed progression, error patterns, and accuracy trends over time to help users monitor improvement.
6. **Visual and Audio Feedback:**
    - Add sound effects for correct or incorrect typing and visual effects for milestones to make the game more interactive and engaging.
7. **Mobile Compatibility:**
    - Develop a mobile-friendly version or app to allow users to practice typing on smartphones or tablets.
8. **Integration with Leaderboards Online:**
    - Enable global online leaderboards where users can compare scores with players worldwide, promoting a community-driven experience.

8. Conclusion

The KeyMaster: A Graphical Typing Challenge successfully provides an interactive platform for users to improve their typing skills, focusing on both speed and accuracy. Through real-time feedback, a high score leaderboard, and a user-friendly interface, the game motivates users to practice regularly and track their progress. Testing showed that users of varying skill levels could engage with the game effectively, achieving measurable improvements in typing performance.

The project demonstrates the practical application of programming concepts, data handling, and GUI design in creating an educational tool. While the current version is functional and effective, the proposed future enhancements, such as timed challenges, difficulty levels, and multiplayer modes, have the potential to further enrich the learning experience. Overall, the KeyMaster: A Graphical Typing Challenge is a successful blend of learning and entertainment, offering a foundation for continued development and skill improvement.

9. References

1. Ray, S. (2020). *C++ GUI Programming with SFML*. TechPress.
2. Stroustrup, B. (2013). *The C++ Programming Language* (4th Edition). Addison-Wesley.
3. Free Typing Lessons. (2025). *Typing Speed and Accuracy Improvement Techniques*. Retrieved from https://www.freetyping.org
4. GeeksforGeeks. (2025). *Implementing High Score Leaderboard in C++*. Retrieved from https://www.geeksforgeeks.org
5. Microsoft Docs. (2025). *Handling User Input in C++ Applications*. Retrieved from https://learn.microsoft.com