National University of Computer and Emerging Sciences
Islamabad Campus

CY2001

Data Structures

Project

Section B

Submitted by: Furqan Haider & Muhammad Hadi Waqar
Roll number: 23i-2123 & 23i-2101
Date: 11-05-2025

Table of Contents

Introduction 3
Data Structures 4
Challenges 4
Workflow Diagram 5
Work Distribution 5
Screenshots or Sample Outputs **7**
Conclusion **16**
Appendix **16**

Project

Introduction

The implementation of this project involves the making of a modernized and improved version of the classic arcade-style Xonix game built using C++ and the SFML graphics library. The game expands the scope of traditional gameplay mechanics to include a robust set of features that enhance player experience and system architecture. Players are presented with either single-player, multiplayer, or ranked modes, which challenge and excite players from different perspectives. Players will also be able to save and load their game progress, manage their profiles, and connect with one another through an integrated friend system.

These all combined offer efficient operations, such as theme management, leaderboard ranking, friend requests, user authentication, etc. The project makes use of advanced data structures: AVL Trees, Heaps, Hash Tables, and Linked Lists. All these have to support a rich user-satisfying graphical interface, which is intuitive and engaging owing to dynamic menus, selection of themes, and in-game animations.

This project is an elaboration of the application of data structure knowledge to real-life software development beyond the stage of gameplay. It touches numerous aspects such as modular design, file manipulation, user interaction, and persistent data maintenance, thereby making it a fun game with technical appeal.

Key Features:

- Single Player, Multiplayer, and Ranked Match Modes
- Save and Load Game functionality
- Dynamic Leaderboard with Heap structure
- Friend System using Linked Lists
- AVL Tree for Theme Inventory
- Hash Table and AVL for User Management

Tools Used:

- Language: C++

- Library: SFML

Development Environment: Visual Studio Code

Data Structures

Data Structure	**Use Case**	**Justification**
AVL Tree	Theme selection and Player BST	Fast search, insertion, and balancing for UI performance
Linked List	Managing friends and pending friend requests	Dynamic memory use and simple traversal
Hash Table	Username lookup during login/registration	Constant time search
Heap	Leaderboard to maintain top scores	Efficient max retrieval and sorting
Structs	TileNode, Enemy, PlayerInfo, etc.	Simple grouping of related data

Challenges

- Managing consistent visual themes across different modes
- Project flow was disrupted when it became difficult to manage and link a large.cpp file with generated files because header files were missing.
- Distraction of War.
- Distraction of ongoing PSL.
- Implemented a global AVL-based theme system ensuring all modes fetch current theme state.
- Handling dynamic friend relationships
- Used linked lists for storing friend and pending request data, with persistent file storage.
- Encountered delays in resuming the game due to slow save/load serialization of a large grid from a text file.

Workflow Diagram

Keeping in mind the available timeframe of two weeks, the project was executed in an intense and structured mode, laying emphasis on faster development and collaborative integration of features.

Week 1: Planning and Core Feature Development

- Finalized game design and user requirements
- Set up SFML environment and graphics assets
- Implemented login/registration system
- Developed basic game loop and single player mode
- Integrated theme system using AVL Trees

Week 2: Feature Expansion and Testing

- Added multiplayer and ranked match functionality
- Implemented save/load game state system
- Developed leaderboard using heap structure
- Built friend system using linked lists and hash table
- Conducted testing, debugging, and visual polish

Key Milestones:

- Login and user system complete by mid-Week 1
- Single player and theme feature by end of Week 1
- Multiplayer, leaderboard, and friend system completed by mid-Week 2

Full testing and refinement done by project deadline

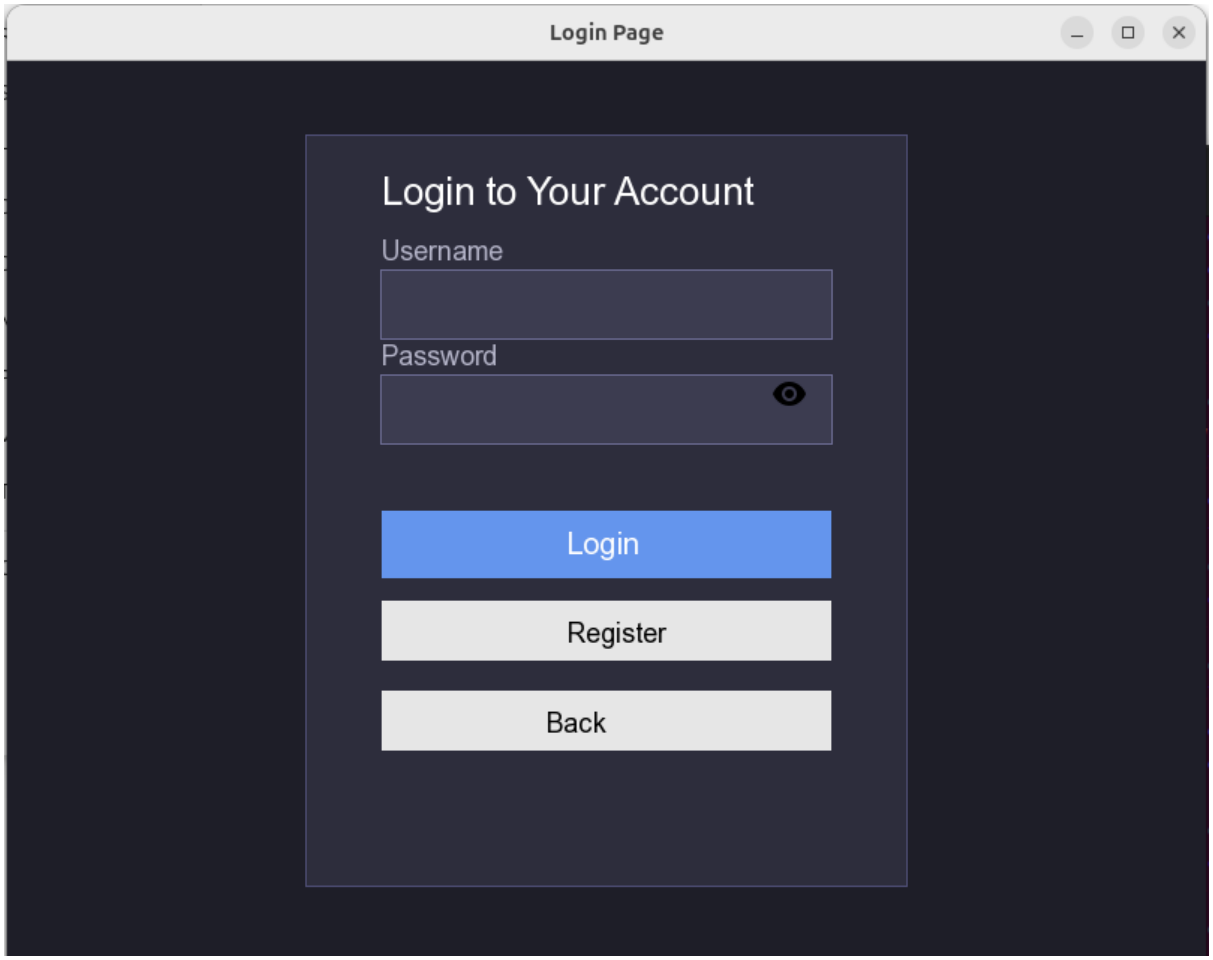
Work Distribution

Feature/Module	**Furqan Haider**	**Hadi Waqar**
UI and Menu Navigation	✓	

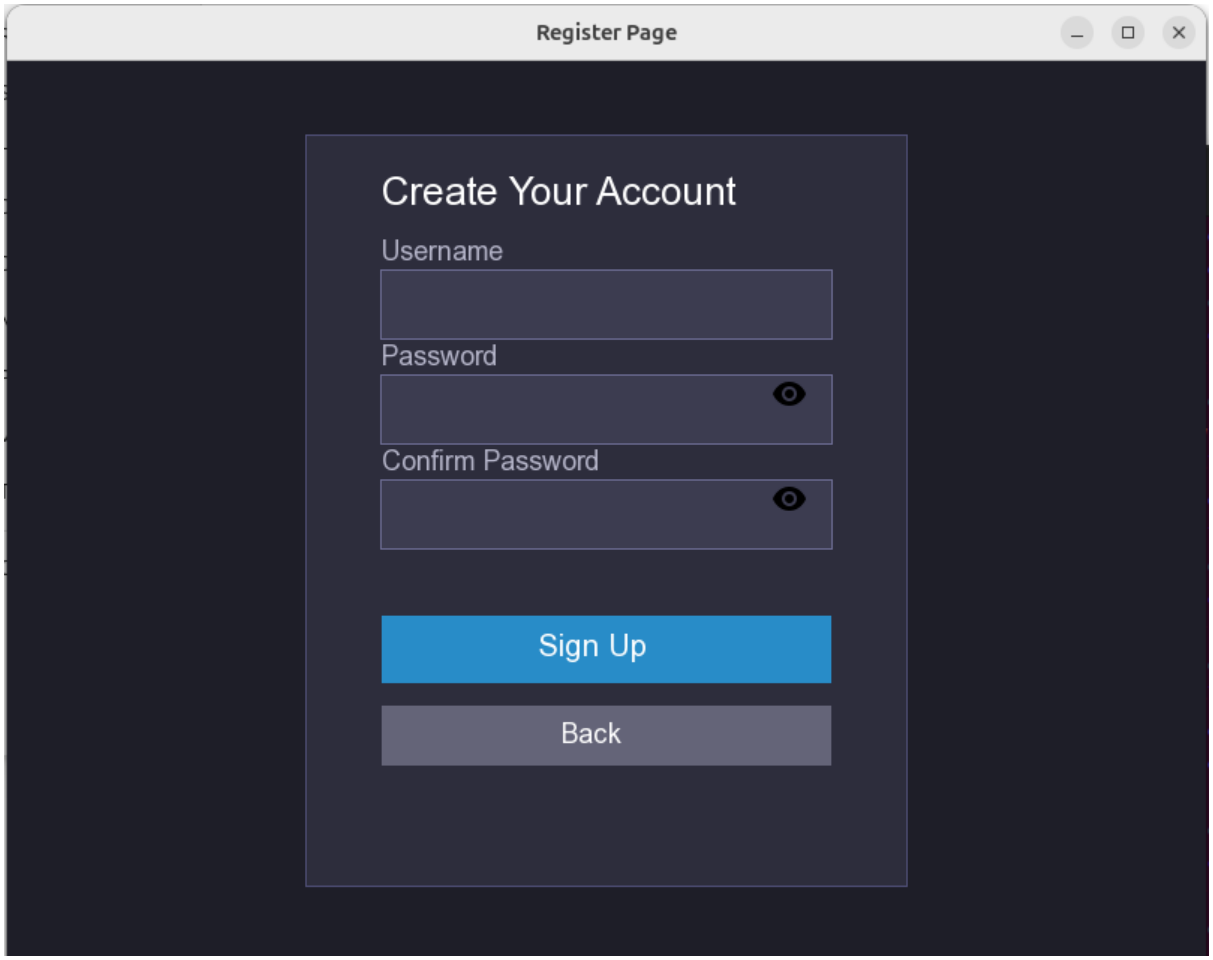
Points System and Power-ups Logic	✓	
Save/Load System	✓	
Audio Features		✓
Theme Selection with AVL Tree	✓	
Social Features (Friends etc.)	✓	
Leaderboard with Heap		✓
User Authentication (Login/Reg)		✓
Testing and Debugging		✓
Matchmaking Queue	✓	
Report Writing		✓

Screenshots or Sample Outputs

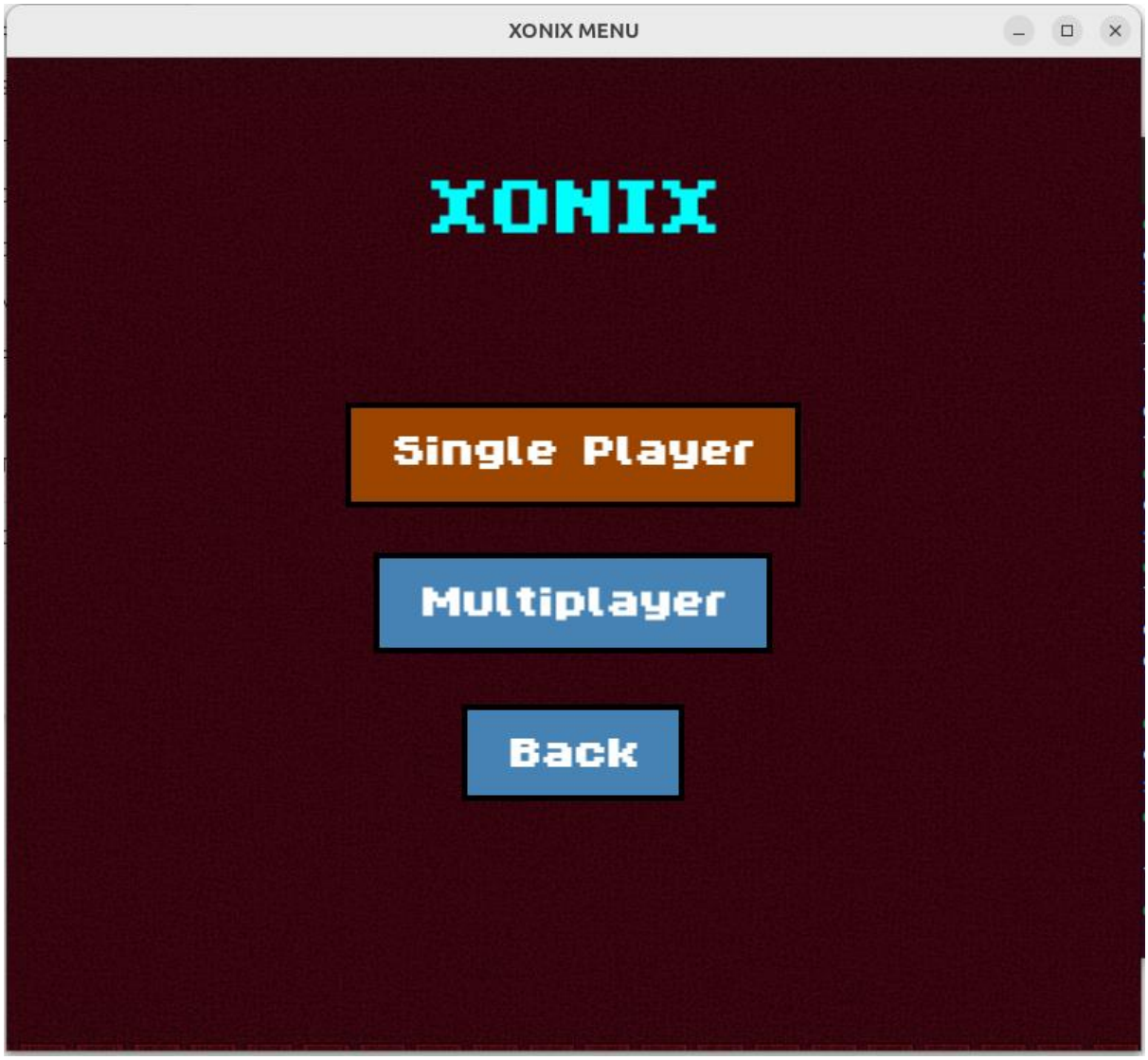
Login Page



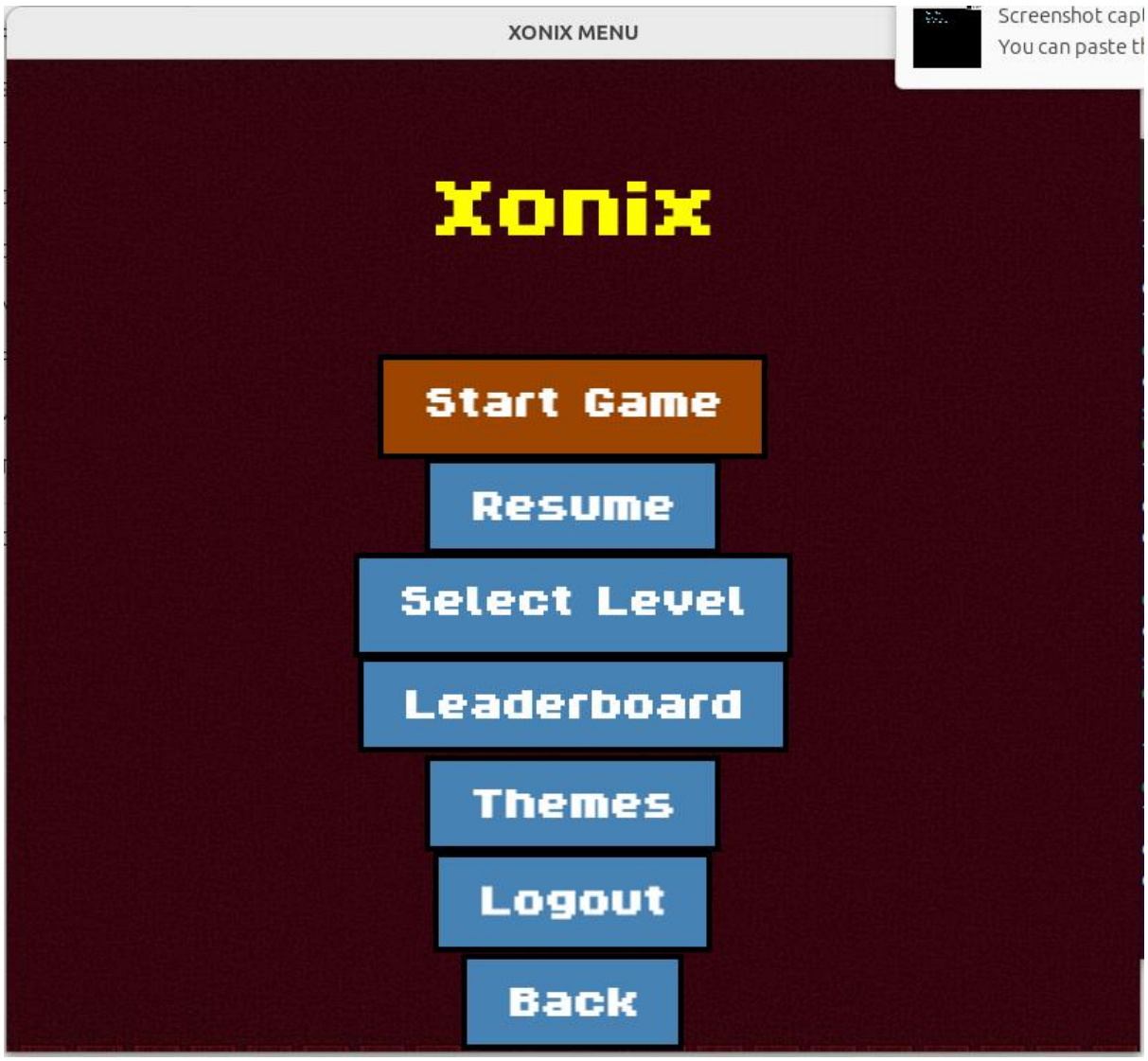
Register Page



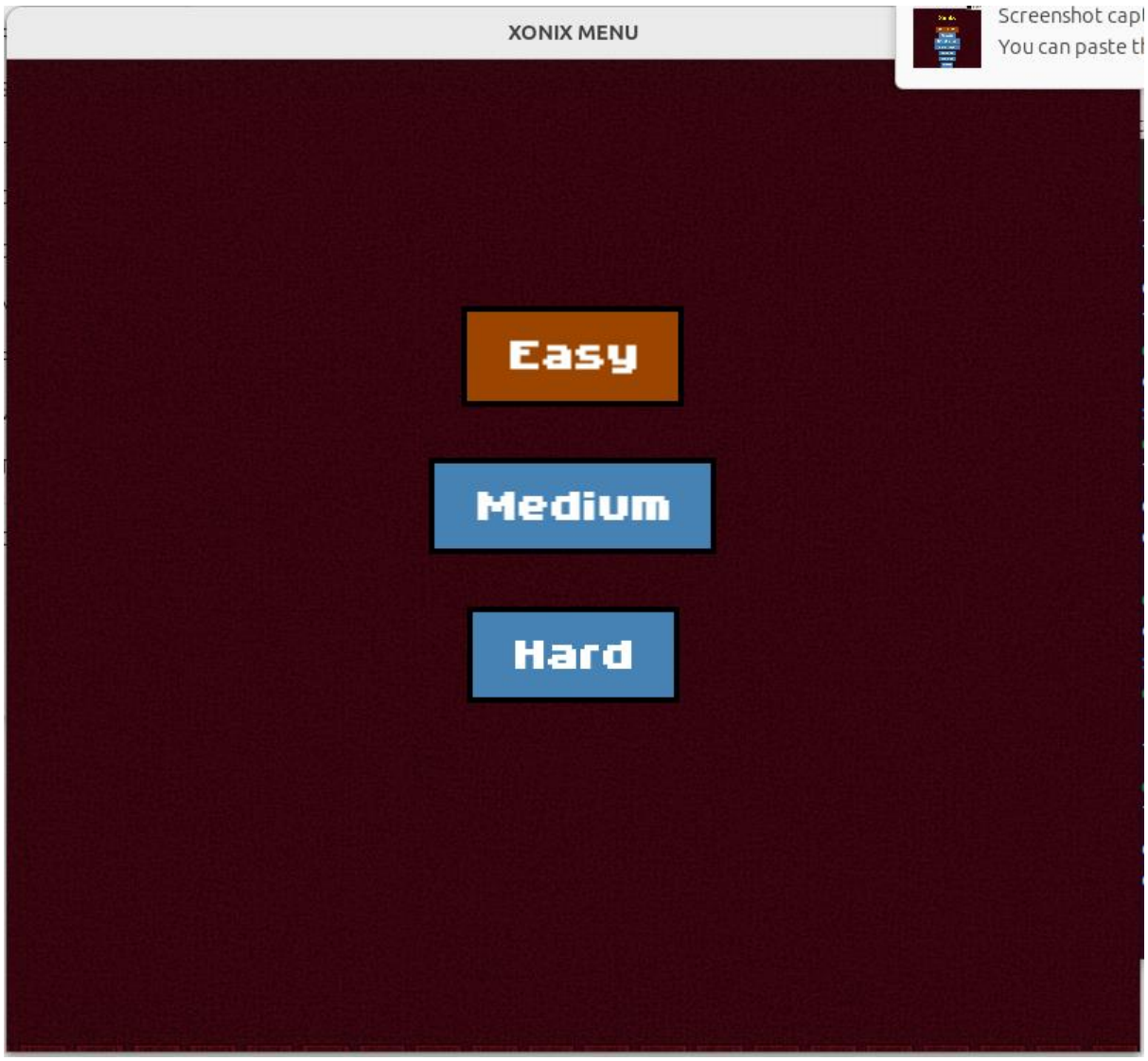
Mode Menu



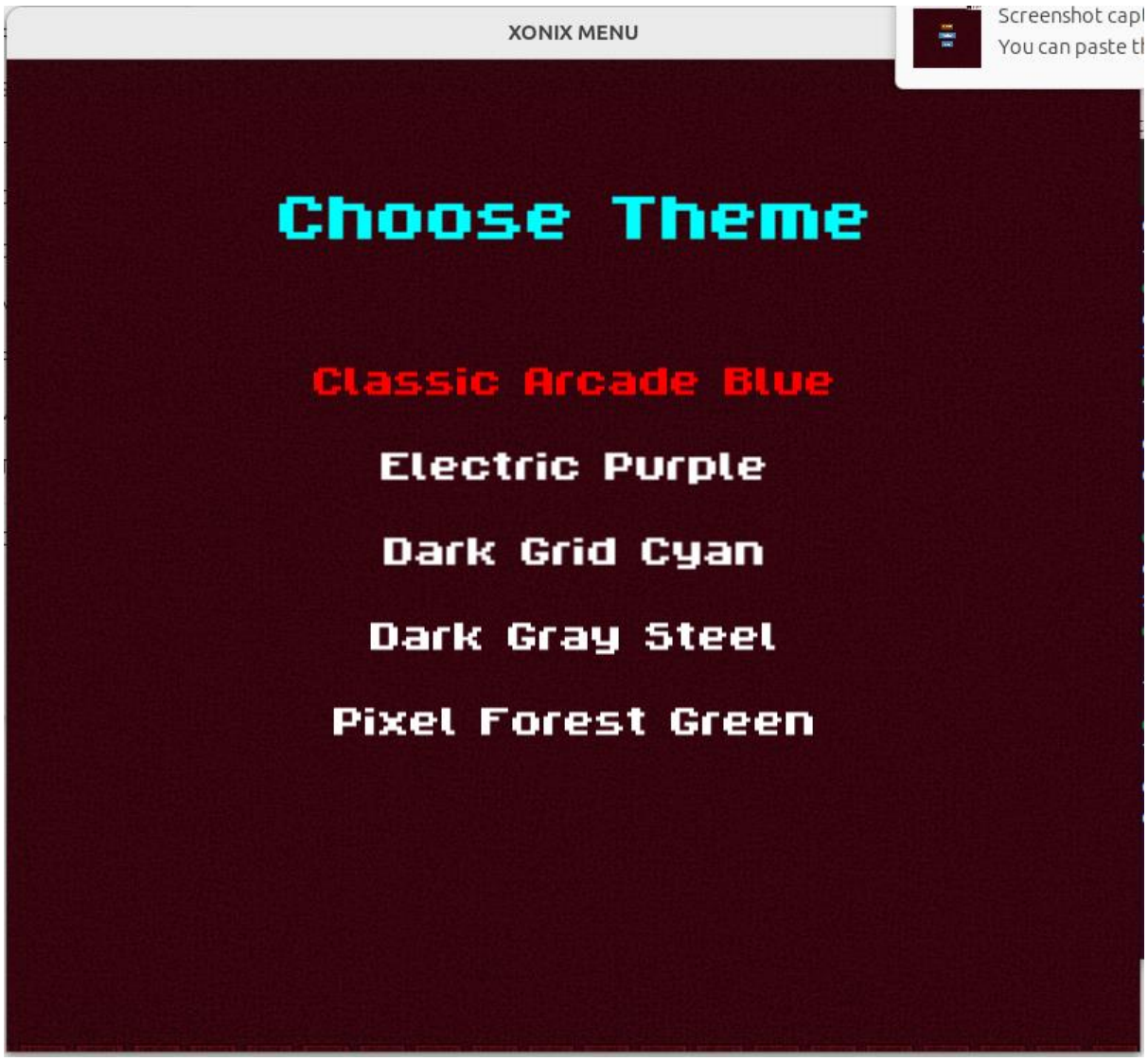
Main Menu


XONIX MENU
Xonix
Start Game
Resume
Select Level
Leaderboard
Themes
Logout
Back

Level Menu

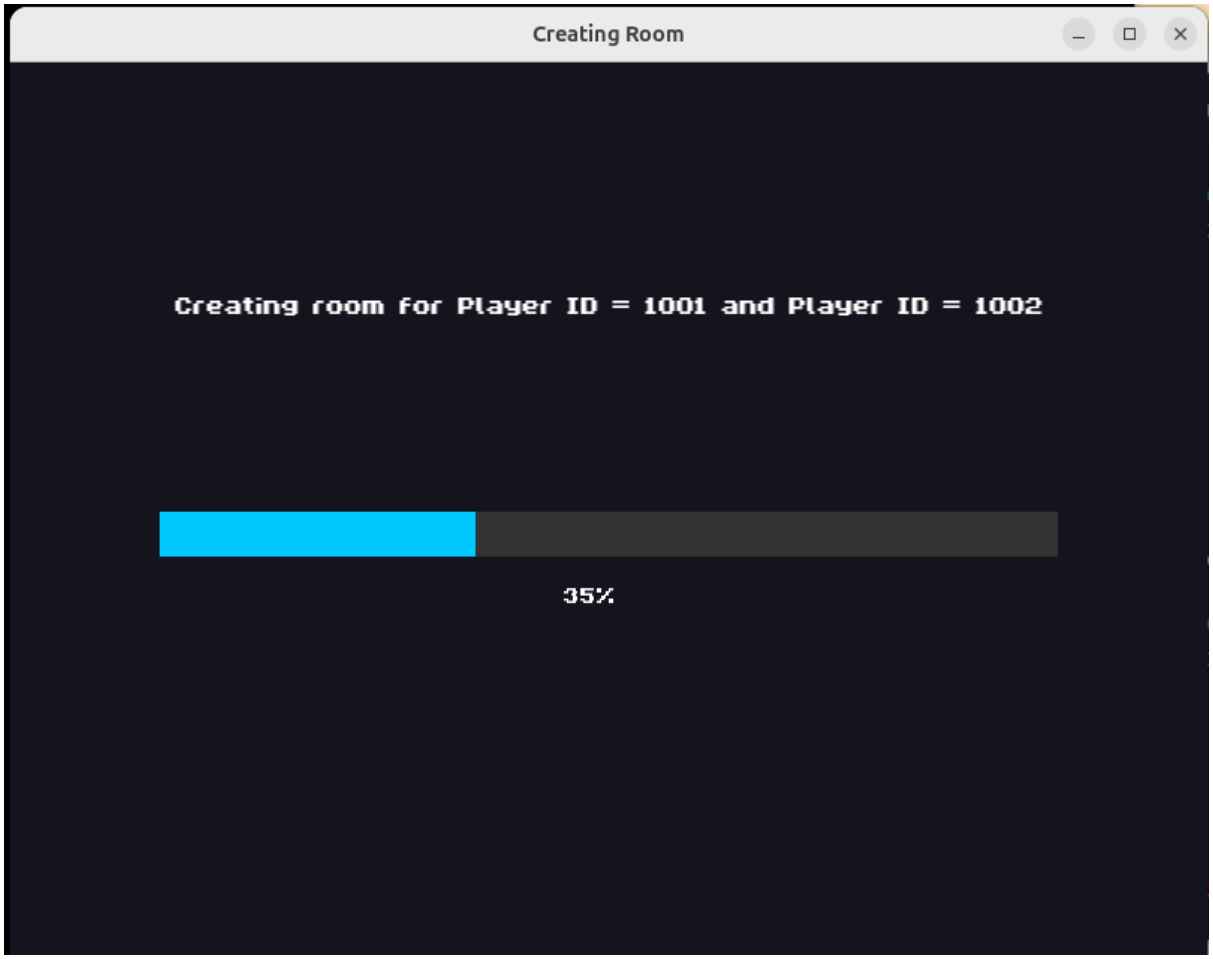


Themes Menu

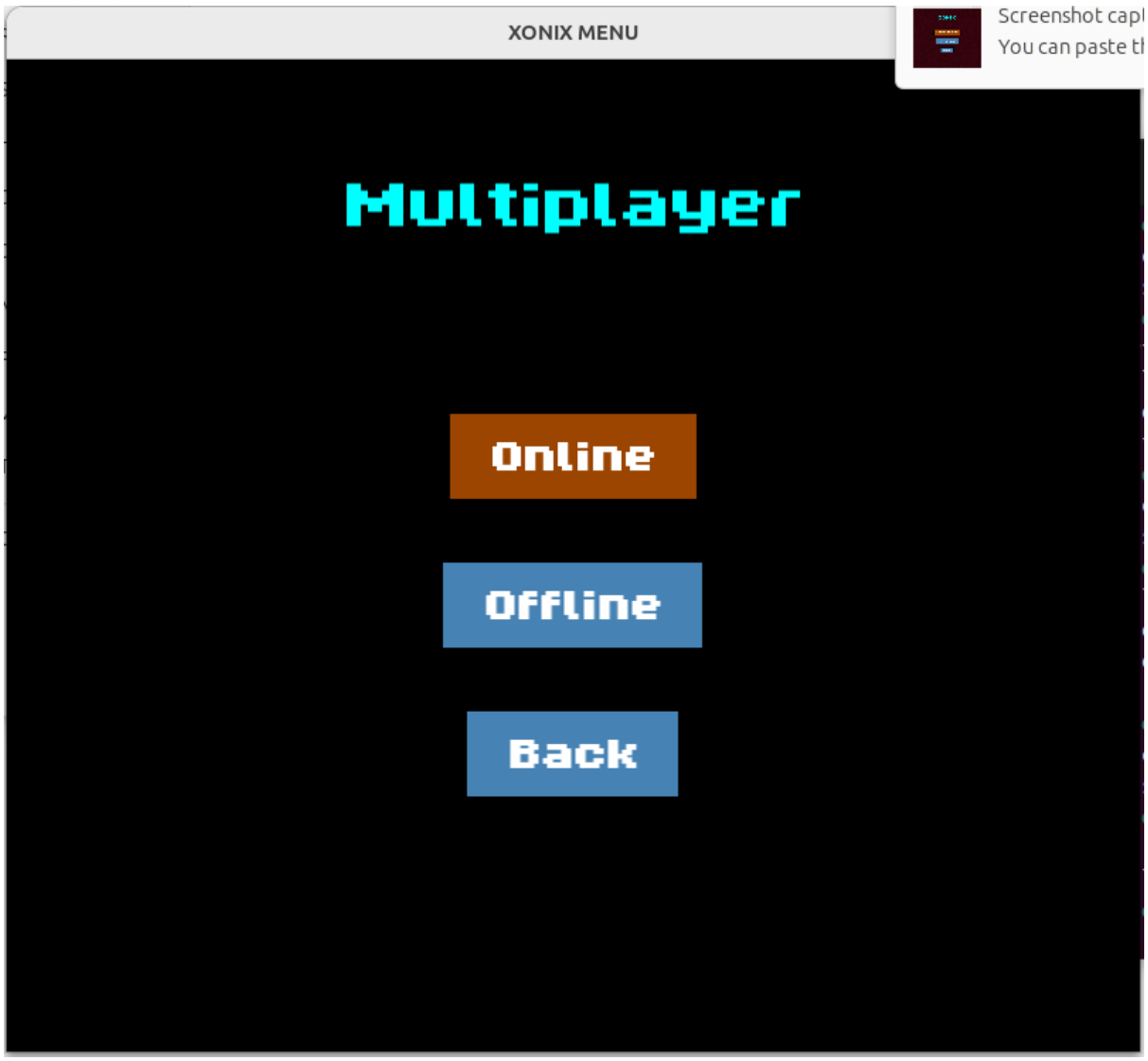

XONIX MENU
Screenshot capt
You can paste th
Choose Theme
Classic Arcade Blue
Electric Purple
Dark Grid Cyan
Dark Gray Steel
Pixel Forest Green

Leaderboard

Room Creation



Multiplayer Menu



Conclusion

This project gave participants practical experience using sophisticated data structures like Min-Heaps, Linked Lists, Hash Tables, and AVL Trees in the context of game development. The effective management of game graphics, user inputs, and state transitions was made possible by the integration of a user interface using SFML. We gained a better understanding of how these data structures can be applied to improve game features and performance during the process. The project's main conclusions and insights are listed below:

- Practical knowledge of sophisticated data structures in the context of game development, including AVL Trees, Hash Tables, Linked Lists, and Min-Heaps.
- Using SFML, a comprehensive user interface was designed and integrated, improving the control of user inputs, graphics, and game state transitions.
- To keep a top-ranking system with real-time updates and guarantee that only the top-scoring players stay on the board, a Min-Heap was implemented for the leaderboard.
- Used an AVL tree for the theme inventory to keep data balanced and searchable, ensuring smooth and responsive UI navigation as new items were added dynamically.
- The ability to link and manage numerous large.cpp and .h files without conflicts, guaranteeing consistency across all features and modes, strengthened my modular programming skills.
- The debugging process improved troubleshooting abilities in complex systems by identifying runtime exceptions, logic errors, and file linking problems.
- Using queues and hash tables, We developed matchmaking, profile systems, and multiplayer features, giving us hands-on experience with dynamic, user-driven systems.

Appendix

https://youtu.be/fwt2jibPeMU?list=PL6xSOsbVA1eb_QqMTTcql_3PdOiE928up

https://youtu.be/esGMreLmed0?list=PLs6oRBoE2-Q_fX_rzraQekRoL7Kr7s5xi

https://learnsfml.com/