The Mysterious Island Adventure Game

(A Text-Based Survival Game)

Md. Rakibul Islam Sagor (ID: 2522028042)
Kaushik Sharma (ID: 2524251642)
Chiranjeet Sarkar Amit (ID: 2523703042)
Md. Jahidul Haque Abir (ID: 2523059642)
Dewan Tahsin Alam (ID: 2524423642)
ECE Department Major in CSE, Course Code: CSE-115, Section: 4, Group: 01 August 11, 2025

❖ Introduction:

The purpose of this report is to document and analyze the final project update for "The Mysterious Island Adventure." The game is a text-based adventure, a genre that relies on narrative and player choice to create an immersive experience. The core objective is to create a compelling story that challenges the player to navigate a dangerous environment, solve puzzles, and ultimately escape the island. This report will dissect the components presented in the update, from the overarching theme to the underlying code, providing a holistic view of the project's progress and its future trajectory.

❖ Project Overview:

The game takes place on a mysterious island. The game begins with the player starting on a beach. Players explore various locations — the jungle, cave, volcano, swamp, and abandoned village — by selecting from menu options. The objective is to uncover clues, collect treasures, and ultimately solve the island's mystery or choose to exit the game.

❖ Summary:

This report provides a detailed analysis of "The Mysterious Island Adventure," a text-based survival quest game, based on the Project Update 1 presentation. The project establishes a compelling narrative foundation centered on survival, exploration, and uncovering ancient mysteries on a deserted island. The initial code, written in C, demonstrates a straightforward, menu-driven architecture that effectively manages game state and player choices for an initial prototype. While the current implementation is limited to a basic main menu and a single sub-location (the jungle), it provides a solid foundation for expansion. The user interface, as shown in the provided screenshots, is clear and functional for a console-based game. This project shows significant promise and, with a structured development plan, has the potential to evolve into a rich, engaging experience. This report will detail the project's strengths and offer a clear roadmap for future development.

❖ Game Concept and Narrative:

The player wakes up on an unfamiliar island. The scene is immediately tense and full of mystery. Key sensory descriptions include:

- The blazing sun
- Crashing ocean waves
- Eerie sounds from the jungle

The narrative hints at:

- A forgotten past or memory loss
- Hidden ancient civilizations
- Whispered legends of temples and treasures
- A lurking sense of danger

This creates an atmospheric and suspenseful environment ideal for a survival adventure game.

The narrative structure, while not fully detailed, is hinted at through the main menu options: exploring the jungle, entering a cave, visiting a volcano, checking a swamp, and visiting an abandoned village. Each of these locations suggests a distinct set of challenges and secrets, providing a natural progression for the story. A well-developed plot arc could involve the player piecing together their lost memory while simultaneously uncovering the island's secrets and working towards a final escape.

❖ Architectural Design and Code Analysis:

📐 Architectural Analysis

1. **Modular Area Design (State Machine Pattern)**
   - Each location (jungle, cave, volcano, etc.) is treated as a "state" using the variable inRoom.
   - State transitions are explicitly handled through updates to inRoom, e.g., inRoom = 2;.
2. **Main Game Loop**
   - Controlled via while (inGame).
   - Allows continuous gameplay until the player exits or wins.
3. **Linear Area Flow Within Loops**
   - Each area is implemented using a while (inRoom == X) loop.
   - Players are presented with a static set of 4 choices per area.
4. **Hardcoded Branching**
   - All possible choices and transitions are statically defined using switch cases.
   - No dynamic or data-driven branching (e.g., external config or JSON files).
5. **No Use of Functions per Area**
   - All gameplay logic is inside main().
   - Could benefit from modularizing each area as a function for better readability.
6. **No Inventory or Persistent State Objects**
   - Player decisions don't affect long-term variables.
   - No inventory, health, or memory of actions across rooms.

👷 Code Analysis

1. **Text Effect Function (typeEffect)**
   - Uses usleep() for timed character-by-character printing.
   - Enhances immersion with a retro/console feel.
2. **Color-coded Output**
   - ANSI escape codes like \033[1;32m are used for visual distinction of menus and narrative.
   - Increases user engagement in the CLI interface.
3. **User Input Handling with scanf()**
   - Simple, direct method for capturing numeric choices.
   - Lacks input validation (e.g., non-integer or buffer clearing).
4. **Structured switch-case Statements**
   - All player actions are managed via switch blocks.
   - Easy to read, but doesn't scale well for larger games.
5. **State Transitions Hardcoded**
   - Room transitions like inRoom = 2; are directly embedded in switch cases.
   - No abstraction or mapping layer (e.g., table-driven or function pointers).
6. **No Dynamic Memory / Data Structures**
   - Program is fully stack-based.
   - No use of arrays, structs, or dynamic allocations.
7. **Game End Conditions**
   - Explicitly handled using inGame = 0; inside win or exit conditions.
8. **Clean Exit and Message**
   - Game concludes with a styled thank-you message using typeEffect().

❖ Challenge Faced:

Input Validation: Ensuring that non-integer or out-of-bound inputs don't crash the game is an ongoing consideration.

Code Modularity: Currently, the logic is linear. Refactoring into functions for each room will improve readability and maintainability.

Scalability: Managing the growing complexity of rooms and choices may require transitioning to a state-machine-based design in the future.

❖ User Interface and Player Experience:

The player experience is currently focused on making choices from a list of options. The descriptive text, such as "You are in the dense jungle. The trees are very tall, and the air is full of mist," sets the scene well and is crucial for a text-based game. The interaction model is intuitive: the player reads the text, considers the options, and types a number to proceed. A potential area for improvement is the depth of the descriptions and the consequences of choices. For example, a choice like "Look for a shelter" currently has a simple, static outcome. As the game evolves, these choices should lead to more dynamic and complex outcomes that influence the player's survival and progress.

❖ Future Development and Project Roadmap:

Enhance Input Handling: Implement a function to validate and sanitize user inputs, ensuring the game handles non-numeric or out-of-range inputs gracefully.

Add Non-Player Characters (NPCs): Introduce characters with whom the player can interact. These NPCs could offer quests, provide clues, or pose as threats.

Implement a Health/Survival System: Introduce a health variable, a hunger variable, and other survival mechanics that add a layer of challenge. This would make choices like "Look for a shelter" or "Find food" more meaningful.

Enhance Code Structure: While the current C code is functional, it can be refactored for better organization as it grows. Using functions for each room or menu would improve readability and maintainability.

❖ References:

[i] D.Ritchie and B.Kernighan, The C Programming Language; 2nd Edition.

[ii] C programming tutorials from https://www.w3schools.com/c/index.php.

[iii] Game concept idea derived from https://www.youtube.com/watch?v=9qSKFazIXbQ.