Dungeon Explorer: Text-Based Strategy Game

Team No :10

Team Members:BT23CSE037,BT23CSE043,BT23CSE044,BT23CSE045,BT23CSE047

---

1. Problem Statement:

Create a text-based dungeon exploration game. The game challenges users to navigate a grid-based dungeon, surviving through traps, collecting treasure, fighting enemies, and levelling up to face a final boss. The player must manage inventory and health while making strategic decisions through a command-line interface.

---

2. Pain Points:

- Lack of engaging terminal-based RPGs with progression systems.
- Inventory/resource management complexity in text-only environments.
- Repetitive or predictable dungeon layouts diminish replay value.
- Need for better feedback and control in turn-based combat mechanics.

---

3. Solution:

A turn-based, event-driven dungeon crawler built in Python that allows players to:

- Navigate through randomly generated 5x5 dungeons.
- Encounter battles, treasure, traps, and a final boss.
- Use and collect items from a persistent inventory system.
- Save and load game progress using file I/O.
- Experience RPG elements like levelling, XP, and stat increases.

---

4. System Design:

- **Dungeon (Level)**: A grid (grid) where special rooms (battle, trap, treasure, exit) are randomly placed.
- **Hero**: Tracks position, stats, XP, and inventory. Can move, fight, and use items.
- **Monster & Boss**: Each has unique stats that scale with dungeon level.
- **Game Loop**: Handles input, rendering, movement, combat, and menus.
- **Event Triggering**: Unique events occur once per room and affect gameplay.

---

5. Database (Files for Persistence):

File Structure:

- savegame.txt: Stores player state including health, XP, inventory, position, and dungeon state.

6. Enhancements:

- Defending reduces damage in combat, adding strategic depth.
- Turn-based battle logic between player and enemies.
- Map printing with emoji-based visualization.
- Final boss choice and branching outcome.

7. Testing (Prototype):

Test Methods:

- Command interface testing for combat, movement, and item usage.
- Manual playthroughs for event triggers.

Results:

- Grid navigation works as expected; traps and treasures trigger once.
- Battle outcomes depend on defense and item strategy.
- Save/load functions restore exact game state.
- Stat updates (XP, level up) validated.

---

8. Challenges & Learnings:

Challenges:

- Managing nested data for saving/loading game state.
- Balancing randomness with fairness (trap/battle distribution).
- Creating meaningful inventory items and interactions.

Learnings:

- Efficient data structures for real-time updates.
- Design Strategy for grid implementation

---

9. Future Improvements:

- Introduce item crafting and upgrading systems.
- Turn game into a web app using Flask or Django.
- Add user authentication and high-score leaderboard.

10.Creativity Ideas:

Visual Emoji Grid Representation

- The dungeon grid is visually represented with **emojis** (e.g., 🧍 for hero, 🚪 for exit), enhancing user engagement and giving it a playful, interactive feel.

Multiple Dungeon Levels with Scaling

The player can progress across multiple dungeon levels, and each one has new challenges — a mini campaign system!