Design Document

Version 1.0 – October 3, 2024

Project Title

CPSC-2720- Fantasy themed Text-based adventure game

Team members:

Owen Rose

Siem Debesay

Tien Dat La

Isaac Bean

Reporter: Owen Rose

Table of Contents

Introduction

Project Overview:

Environments (Locations)

- Village of Luminara: Starting point; receive the main quest.
- Whispering Woods: Solve Gorwin's Riddle; obtain the Enchanted Map.
- Crystal Caves: Light Reflection Puzzle; receive the Crystal Lens.
- Forgotten Library: Book Sorting Puzzle; gain the Ancient Tome.
- Echoing Mountains: Echo Puzzle; use the Echo Crystal.
- Shadow Marshes: Illusion Maze Puzzle; avoid the Shadow Figure's deception.
- Sanctum of Light: Trial of Wisdom; acquire the Staff of Lumos.
- Malakar's Lair: Final confrontation with Malakar.
- Hidden Grove (Optional): Unlocked by collecting Ancient Runes; offers additional lore.

Characters

- Player character (PC): Protagonist, the player character.
- Elda: Village elder; provides guidance.
- Gorwin: Hermit; challenges with a riddle.
- Thorin: Guardian aiding in the Crystal Caves.
- Lyra: Scholar assisting in the library.
- Mira: Priestess overseeing the Trial of Wisdom.
- The Shadow Figure: Antagonist's emissary; misleads the player.
- Elyndor: Mystic in the Hidden Grove; offers optional quests.

Player Actions

- Movement: go [direction], enter, climb, crawl.
- Environment Interaction: look, examine, search, dig.
- Object Interaction: pick up, use, combine, inspect.
- NPC Interaction: talk to, give [item] to, ask about.

Items

    Usable: Quest Scroll, Enchanted Map, Crystal Lens, Ancient Tome, Herbal Mixture, Staff of Lumos, Echo Crystal, Silver Key.
    Non-Usable with Intricacies:
        Old Coin: Collect three to unlock hidden lore.
        Broken Amulet: Combine with Mysterious Gem for a secret message.
        Mysterious Letter: Decipher with the Cipher Wheel for hints.
        Cipher Wheel: Decodes encrypted messages.
        Ancient Runes: Unlocks the Hidden Grove when all are collected.

Puzzles (7 Total)

    Gorwin's Riddle: Obtain the Enchanted Map.
    Light Reflection Puzzle: Open a hidden passage in the Crystal Caves.
    Book Sorting Puzzle: Reveal the Ancient Tome.
    Echo Puzzle: Replicate sounds to proceed.
    Illusion Maze Puzzle: Navigate the Shadow Marshes.
    Cipher Decoding Puzzle: Decode messages for critical hints.
    Trial of Wisdom: Solve logic puzzles to gain the Staff of Lumos.

Ways to Lose the Game (5 Total)

    Traps from Incorrect Puzzles: Cave-ins, trapdoors, avalanches.
    Getting Lost: Without essential items like the Herbal Mixture.
    Failing Critical Trials: Such as the Trial of Wisdom.
    Deception: Following false guidance from the Shadow Figure.
    Time-Sensitive Failure: Not completing key tasks before certain events (e.g., before an eclipse).

Help System and Error-Checking

    Commands:
        help: Lists available commands and gameplay instructions.
        hint: Provides context-sensitive hints.
    Error-Checking:
        Validates player input to prevent crashes.
        Offers friendly feedback for invalid or nonsensical commands.

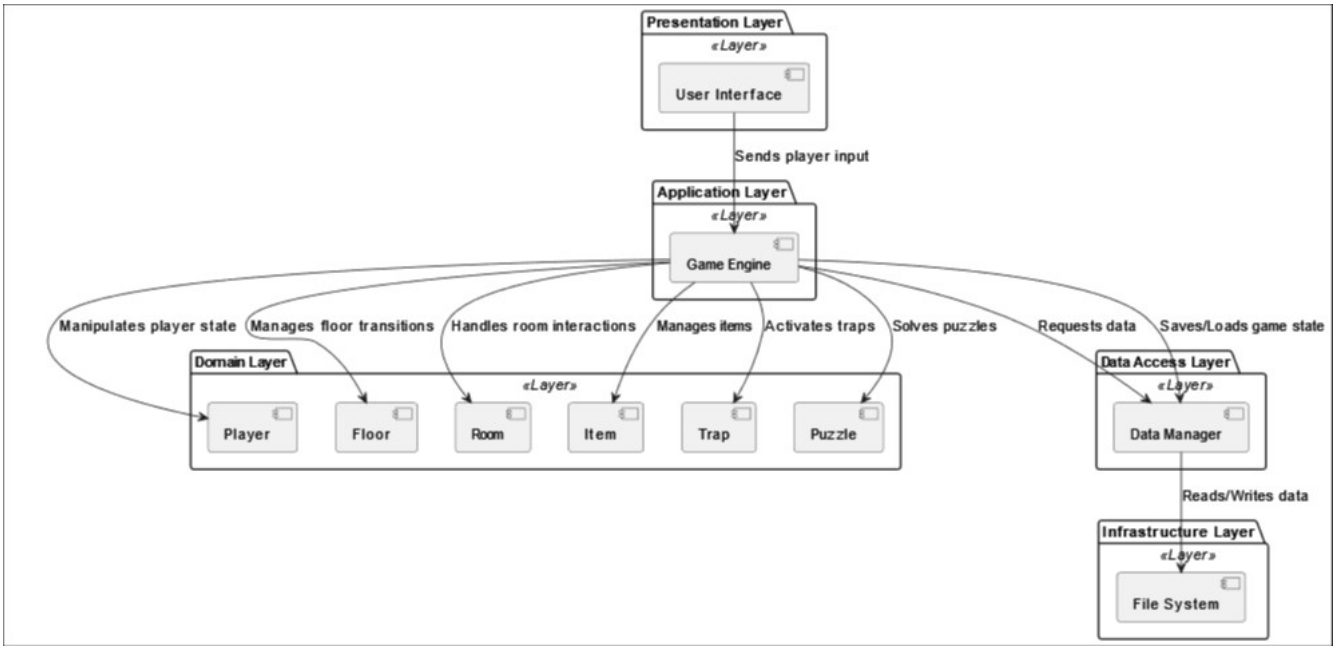
Project Requirements Fulfillment

    Environments: 8+ unique locations with detailed exploration.
    Characters: Multiple NPCs for interaction and plot advancement.
    Actions: Comprehensive commands covering movement, interaction, and puzzle-solving.
    Items: Essential and non-essential items enriching gameplay and lore.
    Puzzles: Seven integral puzzles to challenge the player.
    Lose Conditions: Five ways to lose, adding depth and stakes.
    Help System: Provides guidance and prevents player frustration.
    Error-Checking: Ensures a smooth gameplay experience without crashes.

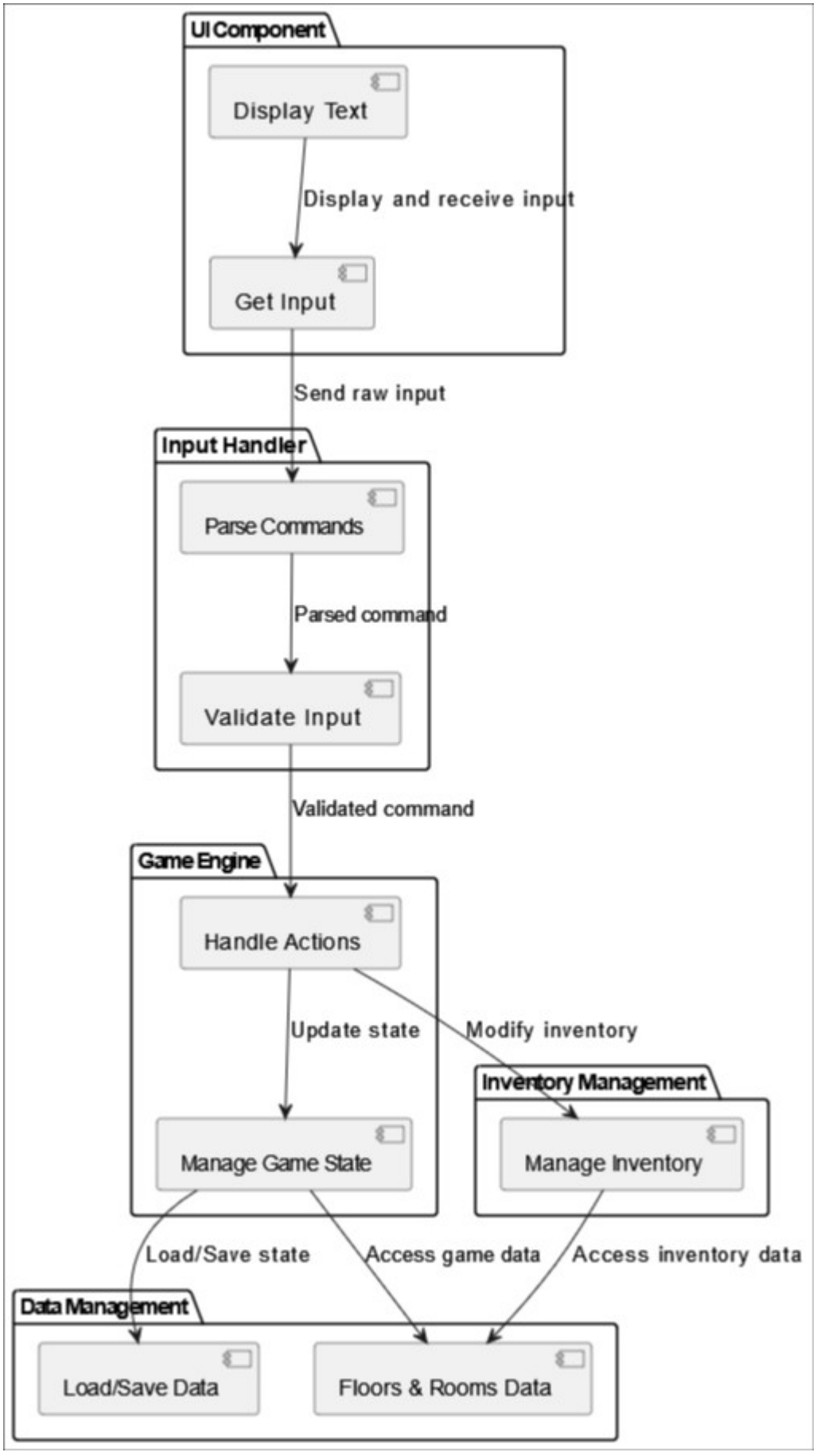
Deliverables:

- System architecture Diagram
- Use case Diagram
- Sequence Diagrams
- Component diagram
- Class Diagram
- Interface prototype
- Raodmap, Gantt chart, timeline

System Architecture



Component Design


UI Component
Display Text
Display and receive input
Get Input
Send raw input
Input Handler
Parse Commands
Parsed command
Validate Input
Validated command
Game Engine
Handle Actions
Update state
Modify inventory
Inventory Management
Manage Game State
Manage Inventory
Load/Save state
Access game data
Access inventory data
Data Management
Load/Save Data
Floors & Rooms Data

Component responsibility outline:

1. User Interface (UI) Component

Display Text:

Displays game information (rooms, items, traps, puzzles) and player status updates.

Interactions: Receives processed data from Game Engine, works with Get Input.

Get Input:

Captures and validates player inputs (e.g., movement, actions).

Interactions: Sends input to Input Handler, works with Display Text for prompts.

2. Input Handler Component

Parse Commands:

Converts raw player inputs into actionable commands.

Interactions: Receives input from Get Input, sends parsed commands to Validate Input.

Validate Input:

Ensures commands are valid within the game context.

Interactions: Receives parsed commands from Parse Commands, sends validated commands to Game Engine, may send errors to Display Text.

3. Game Engine Component

Manage Game State:

Maintains and updates player status, location, and progression.

Interactions: Works with Domain Entities, communicates with Data Management, updates Display Text.

Handle Actions:

Executes player actions (movement, item interactions, puzzles).

Interactions: Receives commands from Input Handler, updates Domain Entities, modifies inventory via Inventory Management.

4. Inventory Management Component

Manage Inventory:

Manages player’s inventory (add, remove, list items).

Interactions: Accessed by Game Engine, may interact with Data Management for item details.

5. Data Management Component

Floors & Rooms Data:

Stores floor, room, item, and puzzle data, updates dynamically.

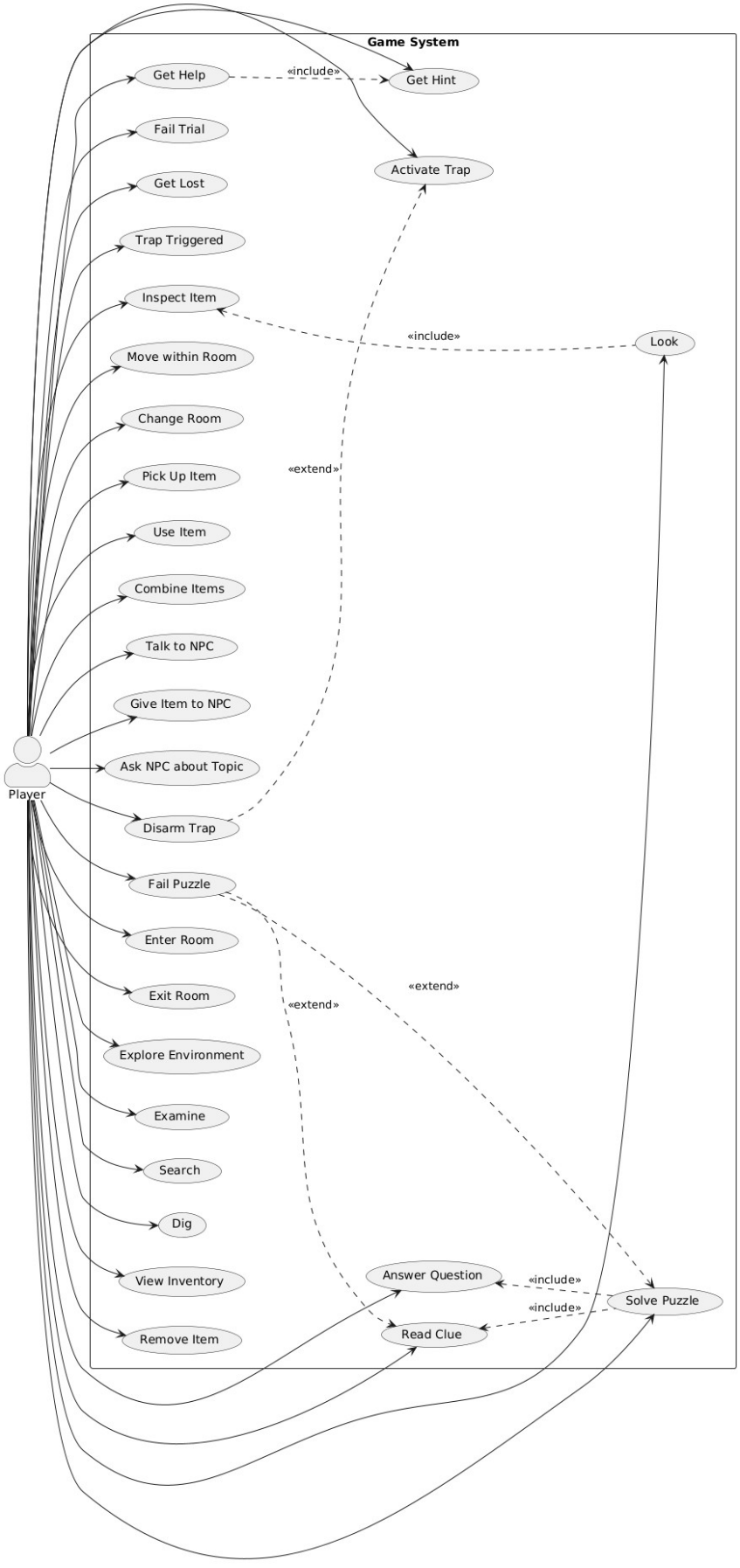
Interactions: Serves data to Game Engine, interacts with Inventory Management, accesses File System.

Load/Save Data:

Handles game state persistence (saving/loading progress).

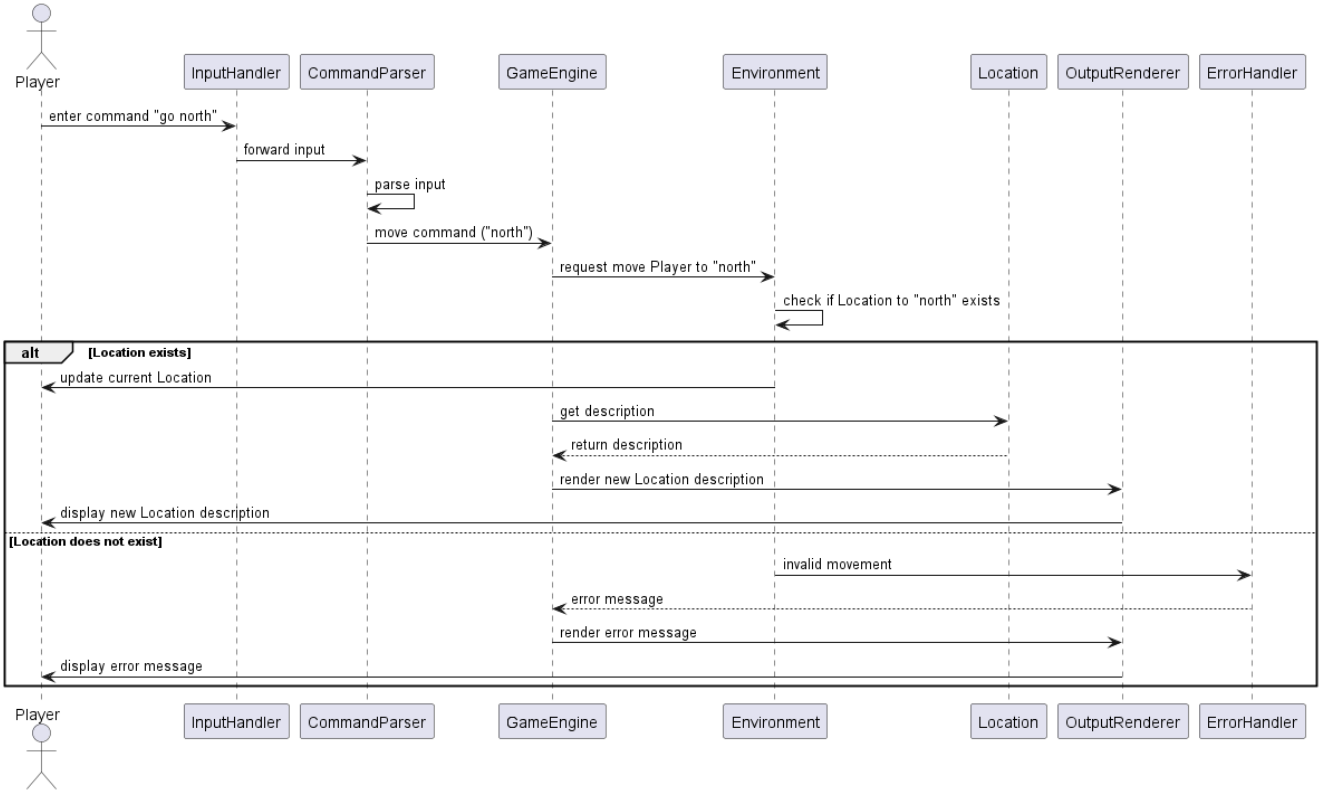
Interactions: Works with Game Engine, uses File System for save files.

Use case Diagram

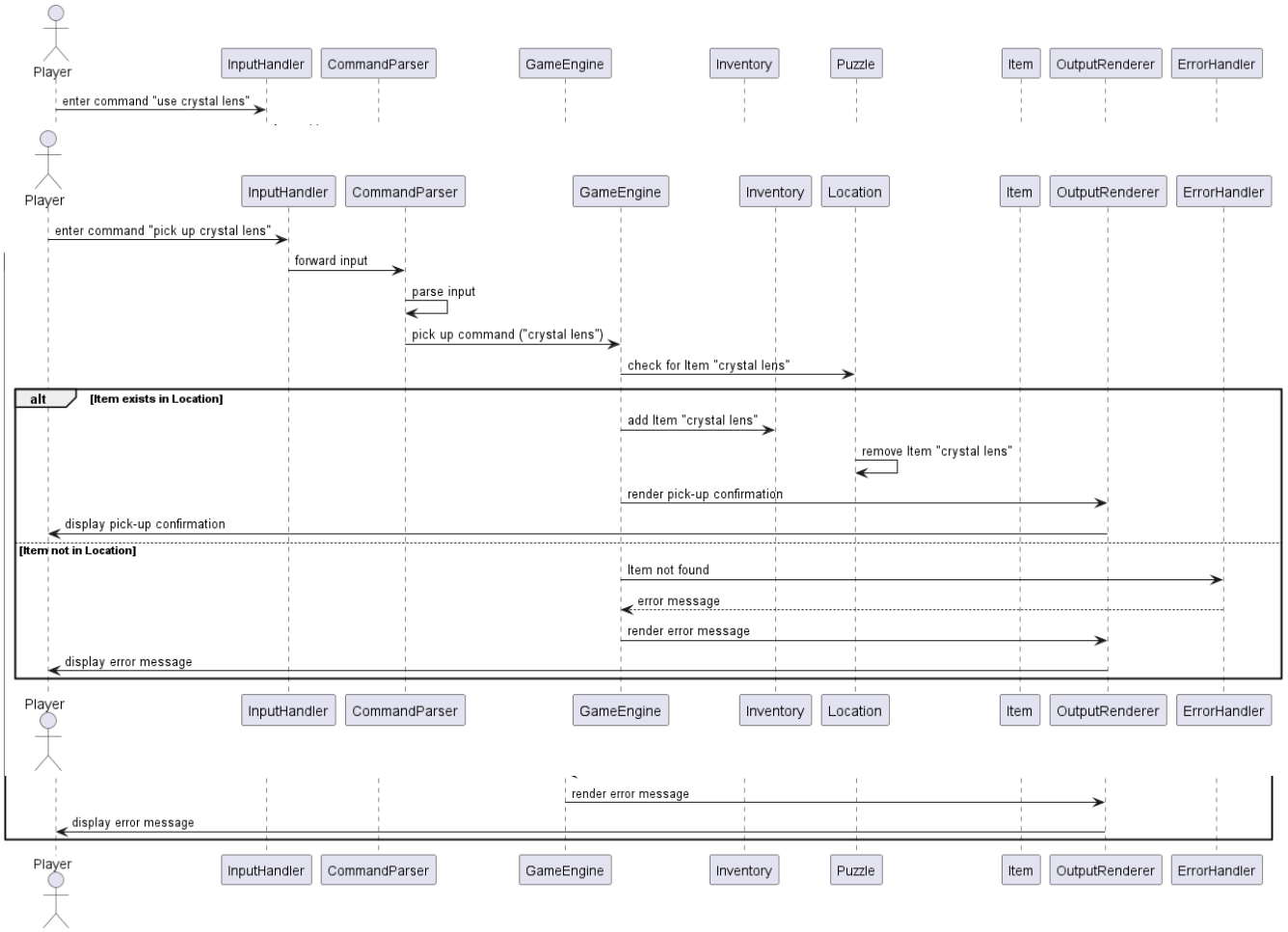

Game System
Get Help
«include»
Get Hint
Fail Trial
Activate Trap
Get Lost
Trap Triggered
Inspect Item
«include»
Look
Move within Room
Change Room
«extend»
Pick Up Item
Use Item
Combine Items
Talk to NPC
Give Item to NPC
Player
Ask NPC about Topic
Disarm Trap
Fail Puzzle
Enter Room
«extend»
«extend»
Exit Room
Explore Environment
Examine
Search
Dig
View Inventory
Answer Question
«include»
Solve Puzzle
«include»
Remove Item
Read Clue

Sequence Diagram

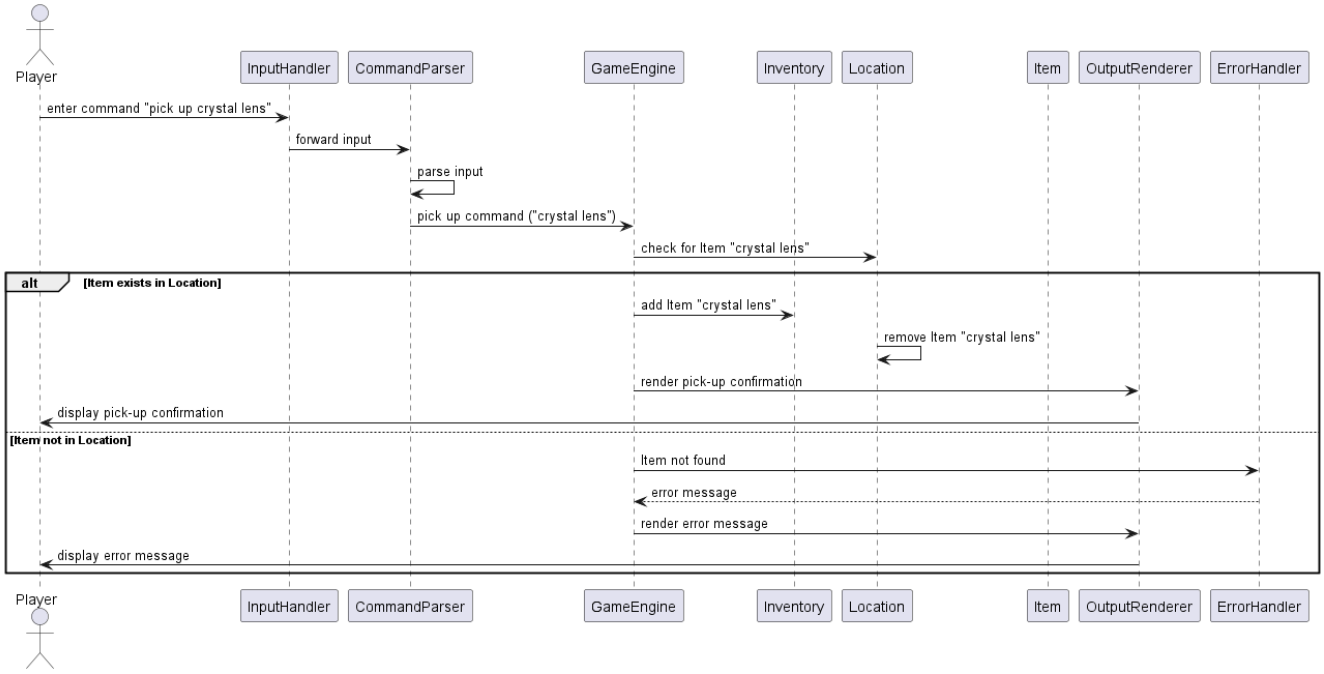
Player moves to new location



Player solves puzzle using item

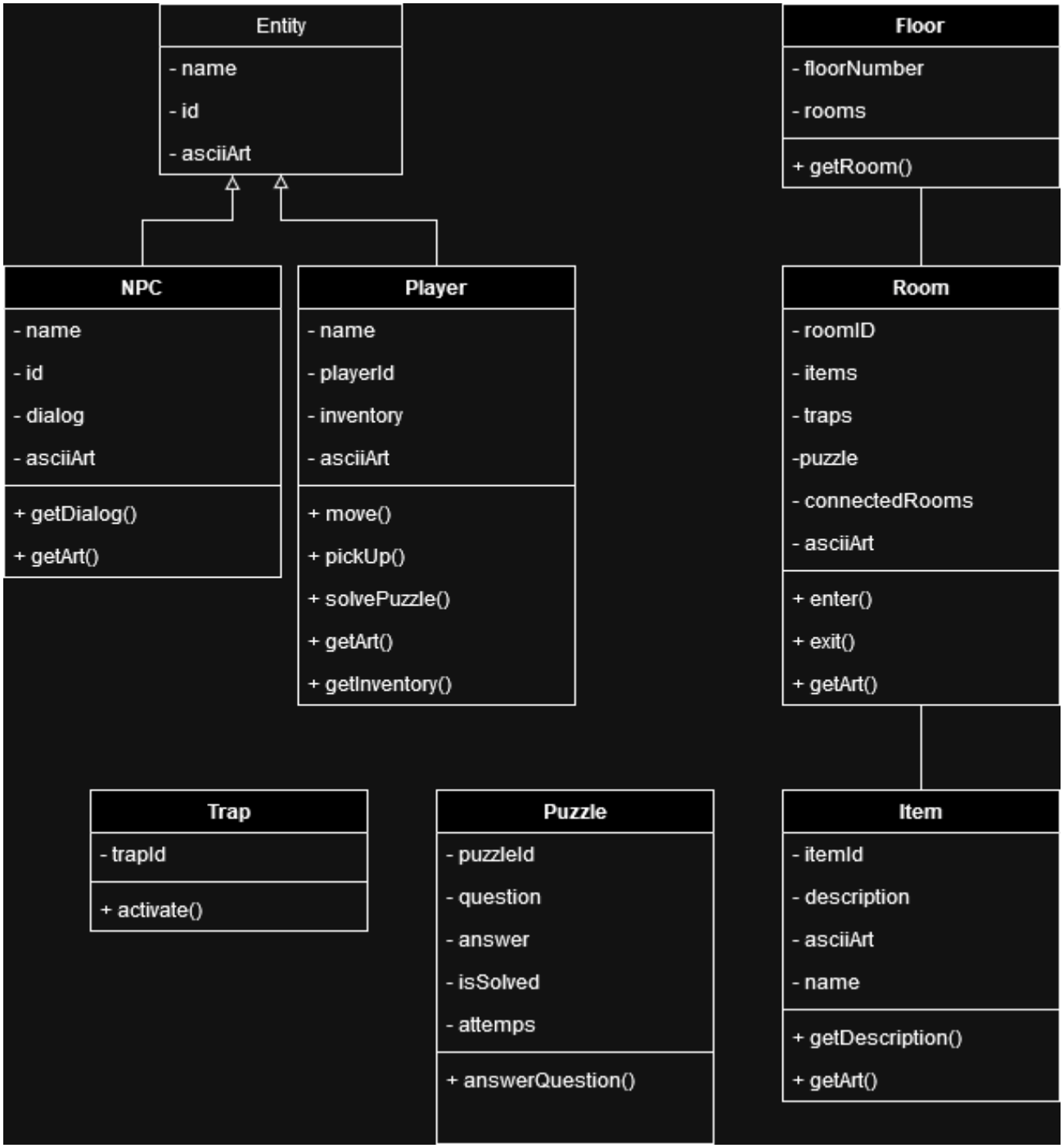


Player picks up an item



Class + Object Design

Class Diagram


Entity
- name
- id
- asciiArt
Floor
- floorNumber
- rooms
+ getRoom()
NPC
- name
- id
- dialog
- asciiArt
+ getDialog()
+ getArt()
Player
- name
- playerId
- inventory
- asciiArt
+ move()
+ pickUp()
+ solvePuzzle()
+ getArt()
+ getInventory()
Room
- roomID
- items
- traps
-puzzle
- connectedRooms
- asciiArt
+ enter()
+ exit()
+ getArt()
Trap
- trapId
+ activate()
Puzzle
- puzzleId
- question
- answer
- isSolved
- attemps
+ answerQuestion()
Item
- itemId
- description
- asciiArt
- name
+ getDescription()
+ getArt()

User Interface Design

Interface Description for the Text-Based Adventure Game

The text-based adventure game will run entirely in a terminal environment, providing players with a command-line experience. The interface is designed to be user-friendly, with a menu system for navigating different options and game commands. There will be occasional use of ASCII art to key moments in the game, but the primary focus remains on text-based interaction.

1. Interface Layout:

The game’s interface is divided into several sections displayed sequentially in the terminal. Players interact with the game world through a combination of text commands and menu-driven options.

Game Title and Introduction Screen:

When the game starts, players are greeted with a title screen, followed by an introductory message that sets up the story and explains the context of the player's quest.

Occasionally, basic ASCII art will be displayed to represent key locations or events.

Gameplay Screen:

The main gameplay screen is text-based, displaying the current location, environment description, and any notable objects or NPCs in the area.

Players issue commands directly at a prompt (e.g., move forward, talk to NPC, pickup item, use item…) to interact with the game world.

The response area shows feedback for the player’s actions and any results of their interactions.

Menu System:

The menu system can be accessed anytime during gameplay by typing a special command (e.g., menu or M or Esc).

Menu Options:

View Inventory: Displays all items the player has collected and their descriptions.

Current Objective Status: Shows the player a reminder what they need to do immediately to progress in the game.

Hint: Gives the user hints on accomplishing the current objective.

Help: Provides a list of available commands and guidance on using the interface.

Exit Game: Allows the player to quit the game at any point.

2. Command-Based Interaction:

Players interact with the game world using a predefined set of commands entered at the prompt. The command structure follows a simple verb-noun pattern (move forward, pick up sword, talk to merchant). Common commands include:

Movement: move forward, enter [location]

Object Interaction: pick up [item], use [item], inspect [item].

NPC Interaction: talk to [NPC], ask [NPC].

Environment Interaction: look, examine [object], search [area].

3. Feedback and Error Handling:

When the player enters a command, the game processes it and provides feedback in the response area. If the command is successful, the player may receive new information, obtain an item, or trigger a plot event.

Invalid commands or unavailable actions are handled gracefully:

Invalid Command: "I don’t understand that. Type help to see a list of valid commands."

Unavailable Action: "You cannot do that here."

4. Interface Aesthetic:

The game will employ a minimalist aesthetic, relying on concise text descriptions and occasional visual breaks through ASCII art. The art will be used sparingly, serving as a way to highlight critical moments or provide a sense of place.

Example Uses of ASCII Art:

Illustrating unique locations (e.g., the entrance to a cave or a mystical artefact).

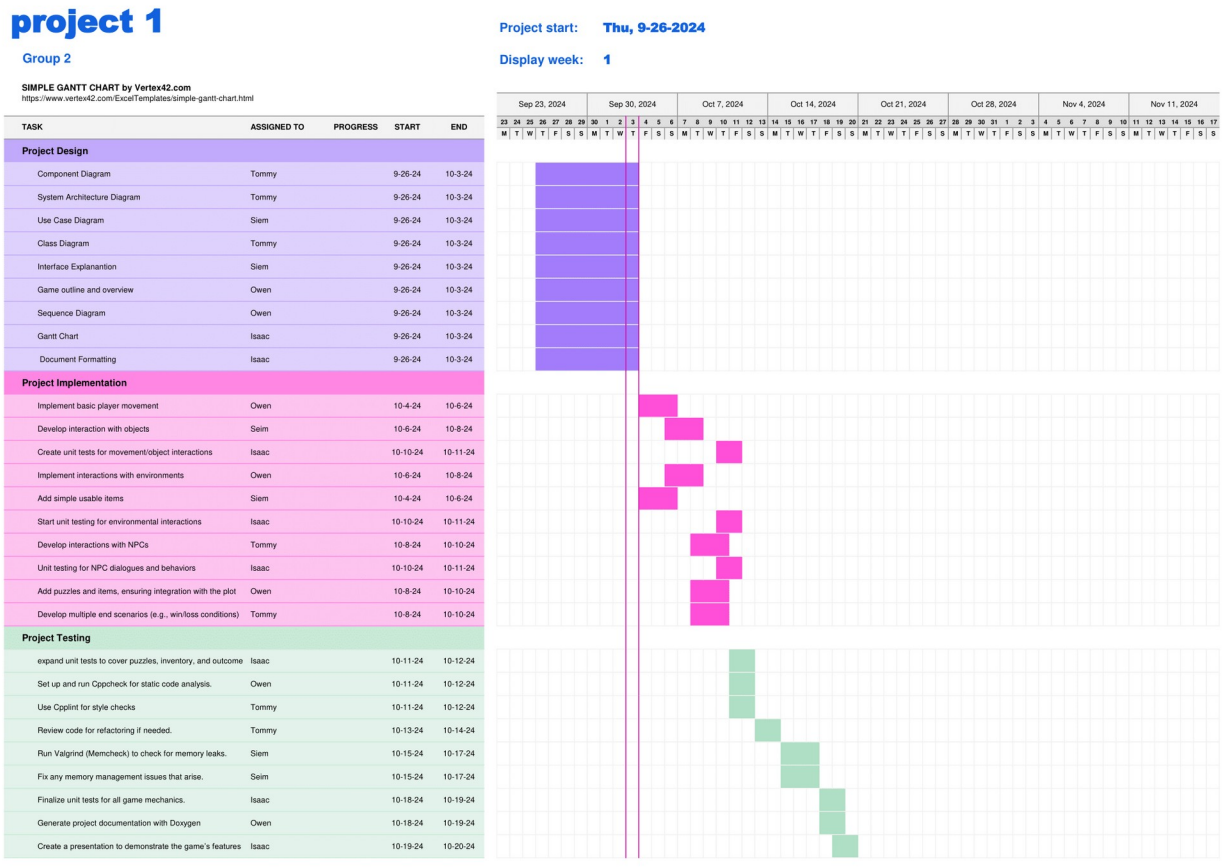
Representing significant NPCs during dialogue.

Displaying items of special importance, like the “Staff of Lumos” or the “Ancient Tome.”

5. Terminal-Exclusive Design:

The game is designed to run exclusively in a terminal, and all interactions are handled through text input. There is no mouse-based interaction, and all navigation is achieved through typed commands.

Plan of Implementation



project 1

Group 2

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project start: **Thu, 9-26-2024**

Display week: **1**

TASK	ASSIGNED TO	PROGRESS	START	END
Project Design				
Component Diagram	Tommy		9-26-24	10-3-24
System Architecture Diagram	Tommy		9-26-24	10-3-24
Use Case Diagram	Siem		9-26-24	10-3-24
Class Diagram	Tommy		9-26-24	10-3-24
Interface Explanantion	Siem		9-26-24	10-3-24
Game outline and overview	Owen		9-26-24	10-3-24
Sequence Diagram	Owen		9-26-24	10-3-24
Gantt Chart	Isaac		9-26-24	10-3-24
Document Formatting	Isaac		9-26-24	10-3-24
Project Implementation				
Implement basic player movement	Owen		10-4-24	10-6-24
Develop interaction with objects	Seim		10-6-24	10-8-24
Create unit tests for movement/object interactions	Isaac		10-10-24	10-11-24
Implement interactions with environments	Owen		10-6-24	10-8-24
Add simple usable items	Siem		10-4-24	10-6-24
Start unit testing for environmental interactions	Isaac		10-10-24	10-11-24
Develop interactions with NPCs	Tommy		10-8-24	10-10-24
Unit testing for NPC dialogues and behaviors	Isaac		10-10-24	10-11-24
Add puzzles and items, ensuring integration with the plot	Owen		10-8-24	10-10-24
Develop multiple end scenarios (e.g., win/loss conditions)	Tommy		10-8-24	10-10-24
Project Testing				
expand unit tests to cover puzzles, inventory, and outcome	Isaac		10-11-24	10-12-24
Set up and run Cppcheck for static code analysis.	Owen		10-11-24	10-12-24
Use Cpplint for style checks	Tommy		10-11-24	10-12-24
Review code for refactoring if needed.	Tommy		10-13-24	10-14-24
Run Valgrind (Memcheck) to check for memory leaks.	Siem		10-15-24	10-17-24
Fix any memory management issues that arise.	Seim		10-15-24	10-17-24
Finalize unit tests for all game mechanics.	Isaac		10-18-24	10-19-24
Generate project documentation with Doxygen	Owen		10-18-24	10-19-24
Create a presentation to demonstrate the game's features	Isaac		10-19-24	10-20-24