**1. Overview**

This project is an **interactive learning game** built using Python and Jupyter Notebook widgets. It allows users to test their knowledge through **quiz questions** and **puzzles** or enjoy short **descriptive stories**. The game features:

* Multiple difficulty levels (**Easy, Medium, Hard**)
* Different game modes (**Quiz, Puzzle, Story**)
* Progress tracking with a **score system**
* Explanations for correct/incorrect answers
* A responsive UI using **IPython widgets**

**2. Features**

**Game Modes**

| **Mode** | **Description** |
| --- | --- |
| **Quiz** | Multiple-choice questions with 4 options |
| **Puzzle** | Riddle-style questions requiring text input |
| **Story** | Displays descriptive text based on keywords (sun, moon, ocean) |

**Difficulty Levels**

| **Level** | **Description** |
| --- | --- |
| **Easy** | Basic general knowledge questions |
| **Medium** | Intermediate-level questions |
| **Hard** | Challenging questions requiring deeper knowledge |

**Additional Features**

✔ **Score Tracking** – Users earn points for correct answers  
✔ **Progress Bar** – Shows completion status  
✔ **Explanations** – Provides reasoning for answers  
✔ **"Play Again" Option** – Restarts the game  
✔ **Responsive UI** – Works well in Jupyter Notebook

**3. Technical Implementation**

**3.1. Dependencies**

* **Python 3.x**
* **Jupyter Notebook**
* Required libraries:

python

Copy

import ipywidgets as widgets

from IPython.display import display, clear\_output

import random

from collections import defaultdict

**3.2. Game Structure**

**Game Data**

* **Questions & Puzzles** (Stored in a list of dictionaries)

python

Copy

questions = [

{

"type": "question",

"question": "What is the capital of France?",

"options": ["Berlin", "Madrid", "Paris", "Rome"],

"correct\_answer": "Paris",

"level": "easy",

"explanation": "Paris has been the capital since 508 AD."

},

# More questions...

]

* **Stories** (Stored in a dictionary)

python

Copy

stories = {

"sun": "The golden sun rose majestically...",

"moon": "The silver moon hung low...",

"ocean": "Endless waves crashed..."

}

**Game State Management**

* Tracks:
  + Current question
  + Score
  + Answered questions
  + Filtered questions based on selection

python

Copy

class GameState:

def \_\_init\_\_(self):

self.reset()

def reset(self):

self.current\_question = 0

self.score = 0

self.game\_state = "setup"

self.answered\_questions = defaultdict(bool)

self.filtered\_questions = []

**Widgets Used**

| **Widget** | **Purpose** |
| --- | --- |
| RadioButtons | Select difficulty & game mode |
| Button | Start, Next, Restart actions |
| Text | Input for puzzles & story keywords |
| IntProgress | Progress bar |
| Output | Display area for questions/results |

**3.3. Key Functions**

| **Function** | **Description** |
| --- | --- |
| filter\_questions() | Filters questions based on level & type |
| start\_game() | Initializes the game |
| check\_answer() | Validates answers & updates score |
| next\_question() | Moves to the next question |
| restart\_game() | Resets the game |
| update\_display() | Handles UI updates |

**4. How to Run**

1. **Install Dependencies** (if needed):

bash

Copy

pip install ipywidgets

1. **Open Jupyter Notebook**:

bash

Copy

jupyter notebook

1. **Copy & Run the Code** in a new notebook cell.
2. **Interact with the Game**:
   * Select **Level** & **Type**
   * Click **Start Game**
   * Answer questions or enter story keywords
   * View score & play again

**5. Future Improvements**

* **Add more questions** (expand the database)
* **Implement a timer** for quiz mode
* **Add images** for visual questions
* **Save high scores** using a simple database
* **Deploy as a web app** (using Voilà or Streamlit)

**6. Conclusion**

This interactive learning game provides an engaging way to test knowledge while improving Python and widget-based UI skills. It is easily extendable with new questions, features, and deployment options.