Documentation

- Instructions to run locally:
  - I have used Visual code's extension named LiveServer using this tutorial it worked pretty well and quickly since I didn't need the privacy part just use http instead of https
  - Any other local server options should work such as Docker and other solutions
  - I tested it on Chrome and Safari on PC, Iphone 13 and Iphone 15, unfortunately i have no android devices for testing
- Assets and Libraries:
  - I've used the following unity packages and other built in solutions:
    - DoTween
    - TMP (included in UI now)
    - JSONUtility
    - Unity Networking
    - Tasks
    - VisualCode for unity

- Overview of code structures and decisions:
  - **Componentes:**
    - StoreManager
      - Initialize the store at the moment just the shelf but can easily be expanded for dynamic shelfs, receives server data (or mock data for testing) and initializes the shelf
    - ServerManager
      - Used by store manager to acquire server data and parse it to classes, for StoreManager to use
      - Holds the Data classes, In a more complex situation where more data manipulation would be required the data classes would be in a different file
      - Can be easily expanded to also update server (might also require changes in server)
    - ShelfManager
      - Initializes the books and position them dynamically
    - Book
      - Holds book data that can be edited and potentially updated to server
      - Holds UI and Visuals and logic for display

- **Structures and Decisions**
    - I chose a bookstore because this allowed me an interesting, interactive and visually appealing design that will be more engaging to the user than a UI heavy option
    - ShelfManager holds and initializes the books, can be easily expanded to get the books data back to StoreManager and to the server, with changes to server data more than one can be displayed (small changes to code will also be required)
    - The book holds most of the actual “Game” logic, and the user engagement is with it most of the code is for user interaction and edit, along with movement to engage the user, with almost the same code and a change of Model the book can be changed to any other product
    - Separated ServerManager from StoreManager for less dependency on the implementation and separation of responsiblities of the server side as well as readability
    - Used data classes to parse server data for readability and reusability throughout the code
    - I used Mock Data to insure I can properly test the “Game” itself without any “noise” from server side, once the implementation was done I integrated with the actual server data, in this case it was simple but in bigger projects this method help to separate responsibilities more clearly even between members of a team, an interface between those sides will help maintain clean code and will help with the integration
    - I didn’t add any confirmation messages since the user can see thor changes immediately and I thought more UI Panels will just distract the user
    - This assignment is a pretty straight forward case but I wanted to work on it as an base code that can be changed and expanded with as little change as possible with defined responsibilities and roles
    - My documentation philosophy (from CLEAN CODE - highly recommended!) is that Class names and function names would be understood without added documentation, so the documentation I added is mostly to explain myself in the exercise but my main objective is that reading the code would be clear as possible from first glance and would hopefully be self explanatory 🙂