Library Management System CLI App Challenge

Objective:

Create a Command-Line Interface (CLI) application to manage a library. The system should allow users to add books, check out books, return books, and view the status of the library. **You can only use built-in libraries for the challenge.**

Requirements:

1. **Add Books**:
    - Each book should have a title, author, and unique ID.
    - Store the books in a list or dictionary.
2. **Check Out Books**:
    - Users should be able to check out a book using the book's ID.
    - Ensure that a book cannot be checked out if it is already checked out.
3. **Return Books**:
    - Users should be able to return a book using the book's ID.
    - Update the status of the book to show it is available.
4. **View Library Status**:
    - List all books with their current status (available or checked out).
    - For checked-out books, show the borrower's name and due date.
5. **Error Handling**:
    - Handle errors such as attempting to check out a non-existent book or returning a book that is not checked out.
6. **Save and Load Data**:
    - Save the library data (books and their status) to a file. (use decorator)
    - Load the data from the file when the program starts.

Instructions:

1. **Project Structure**:
    - Create a Python file named `library_management.py`.
    - Organize your code into functions for each feature (e.g., add_book, check_out_book, return_book, view_status).
2. **Data Structures**:
    - Use dictionaries to store book information and their status.
    - Use a list to keep track of borrowed books and their details (borrower's name, due date).
3. **Main Menu**:

- Display a menu to the user with options to add a book, check out a book, return a book, and view library status.

4. **User Input**:
   - Use input() to get user inputs for various actions.
   - Ensure inputs are validated (e.g., check if a book ID exists before checking out).
5. **Saving and Loading**:
   - Save the library data to a JSON file (`library_data.json`).
   - Load the data from the JSON file at the start of the program.
6. **Program Flow**:
   - The program should loop until the user decides to exit.
   - Provide clear instructions and feedback to the user for each action.

Sample Output:

```
Welcome to the Library Management System

1. Add Book
2. Check Out Book
3. Return Book
4. View Library Status
5. Exit

Enter your choice: 1
Enter book title: The Great Gatsby
Enter author: F. Scott Fitzgerald
Book added successfully!

Enter your choice: 4
Books in the Library:
ID: 1, Title: The Great Gatsby, Author: F. Scott Fitzgerald, Status: Available

Enter your choice: 2
Enter book ID to check out: 1
Enter borrower's name: John Doe
Enter due date (YYYY-MM-DD): 2024-07-10
Book checked out successfully!

Enter your choice: 4
Books in the Library:
ID: 1, Title: The Great Gatsby, Author: F. Scott Fitzgerald, Status: Checked Out by Johr
```

Additional Tips:

- Use functions to keep the code organized and reusable.
- Comment your code to explain the logic behind each function.

Good luck! This project will give you hands-on experience with Python and help you solidify your understanding of various programming concepts.