TEAM 5 BUG REPORT

BUG REPORT FOR THE BASE CODE

This bug report highlights several issues and areas for improvement in the Flask-based library management system code.

**Details:**

1. **Database Configuration:**
   * Issue: The database URI is set to **"sqlite:///data.db"**, but there is no confirmation that the **data.db** database file exists.
   * Suggestion: Ensure the database file exists or provide a mechanism to create it.
2. **Missing Routes:**
   * Issue: Routes **/admin/addbook** and **/admin/returnbook** are defined but lack associated view functions or logic.
   * Suggestion: Implement functionality for these routes or remove them if not needed.
3. **Missing HTML Templates:**
   * Issue: HTML templates such as **"index.html"**, **"login\_admin.html"**, **"login\_user.html"**, **"register\_admin.html"**, **"register\_user.html"**, **"add\_book.html"**, and **"return\_book.html"** are referenced in the routes but appear to be missing.
   * Suggestion: Create these HTML templates in the appropriate directory.
4. **Database Relationships:**
   * Issue: Relationships between models like **Library**, **History**, and **Issued** are defined using foreign keys but not set up using SQLAlchemy's relationships.
   * Suggestion: Define relationships between models for improved querying and navigation.
5. **Form Handling:**
   * Issue: There is no logic for handling form submissions in routes such as registration and book addition.
   * Suggestion: Implement form handling logic using Flask-WTF or Flask-Forms for these routes.
6. **Security:**
   * Issue: User passwords are stored as plain text in the database, posing a security risk.
   * Suggestion: Implement password hashing and salting using Flask-Bcrypt or a similar library to enhance security.
7. **Error Handling:**
   * Issue: There is no apparent error handling for exceptions that may occur during database operations or other parts of the application.
   * Suggestion: Implement error handling with try-except blocks and appropriate HTTP error codes.
8. **Logging:**
   * Issue: Logging is not present in the application, making debugging and monitoring challenging.
   * Suggestion: Add logging functionality to assist with debugging and monitoring.
9. **Configuration Management:**
   * Issue: Configuration variables such as the database URI are not separated into a configuration file.
   * Suggestion: Use a configuration file (e.g., **config.py**) to store sensitive configuration variables.
10. **Unit Testing:**
    * Issue: There is no indication of unit tests to verify the functionality of the application.
    * Suggestion: Consider writing unit tests to ensure various parts of the application work as expected.
11. **Deployment:**
    * Issue: The code does not account for deployment to a production environment.
    * Suggestion: Configure a production-ready database, consider using a production web server (e.g., Gunicorn), and address security concerns for production deployment.

BUG REPORT FOR THE ADMIN FUNCTIONS

The provided code contains several issues, including incorrect usage of SQLAlchemy methods, incorrect function signatures, and potential runtime errors. These issues need to be addressed to ensure the code functions correctly.

**Details:**

1. **Incorrect Usage of .add() and .commit() Methods:**
   * Issue: The code attempts to use **.add()** and **.commit()** methods on model classes (**dbc.Issued**, **dbc.Book**, etc.), which is incorrect.
   * Suggestion: Create instances of the model classes, add them to the session using **session.add()**, and commit the session using **session.commit()**.
2. **Incorrect Function Signatures:**
   * Issue: The functions **issue\_book()**, **return\_book()**, **add\_book()**, and **new\_book()** have incorrect parameters. They expect too many arguments.
   * Suggestion: Modify the function signatures to accept the necessary attributes for creating model instances rather than individual attributes.
3. **Querying Database Issues:**
   * Issue: The code attempts to query the database using incorrect syntax, such as **dbc.Issued.query.add()** and **dbc.History.query.filter\_by()**.
   * Suggestion: Use SQLAlchemy's query methods to properly fetch and manipulate data from the database