```
#constants.py

# Define Log Switch
LOG_SWITCH = True



import logging



logging.basicConfig(level=logging.DEBUG,
                    format="%(asctime)s - %(levelname)s - %(message)s",
                    datefmt="%Y-%m-%d %H:%M:%S",
                    force=True)

# Predefined users and roles
users = [{'user': 'jdoe', 'firstname': 'John', 'lastname': 'Doe', 'role': 'librarian'}]
normal_users = [item['user'] for item in users if item['role'] == 'user']
librarians = [item['user'] for item in users if item['role'] == 'librarian']

def authenticate_user(username):
    """Checks if the user is authenticated"""
    return username in normal_users or username in librarians

def authenticate(func):
    def wrapper(username):
        logging.debug(f"{username} is being verified ....")
        if not authenticate_user(username):
            logging.error(f"Unauthorized user {username} detected")
            raise PermissionError(f"Unauthorized user {username} detected")
        return func(username)
    return wrapper

@authenticate
def add_book(username):
    """Allows a librarian to add a new book"""
    if username not in librarians:
        logging.warning(f"User {username} does not have add book permission")
        raise PermissionError(f"User {username} does not have add book permission")

    book_title = input("Enter book title: ")
    book_author = input("Enter book author: ")
    book_isbn = input("Enter book ISBN: ")

    if LOG_SWITCH:
        logging.info(f"User {username} added book: {book_title}, {book_author}, {book_isbn}")

    # Simulated book addition
```

```
        return {
            "title": book_title,
            "author": book_author,
            "isbn": book_isbn
        }

    @authenticate
    def update_book(username):
        """Allows a librarian to update book information"""
        if username not in librarians:
            logging.warning(f"User {username} does not have update book permission")
            raise PermissionError(f"User {username} does not have update book permission")

        book_isbn = input("Enter book ISBN to update: ")
        new_title = input("Enter new title: ")
        new_author = input("Enter new author: ")

        if LOG_SWITCH:
            logging.info(f"User {username} updated book {book_isbn} to: {new_title}, {new_author}")

        # Simulated book update
        return {
            "isbn": book_isbn,
            "new_title": new_title,
            "new_author": new_author
        }

    @authenticate
    def delete_book(username):
        """Allows a librarian to delete a book"""
        if username not in librarians:
            logging.warning(f"User {username} does not have delete book permission")
            raise PermissionError(f"User {username} does not have delete book permission")

        book_isbn = input("Enter book ISBN to delete: ")

        if LOG_SWITCH:
            logging.info(f"User {username} deleted book with ISBN: {book_isbn}")

        # Simulated book deletion
        return {
            "isbn": book_isbn
        }

    @authenticate
    def view_book(username):
        """Allows a user to view book information"""
        book_isbn = input("Enter book ISBN to view: ")

        if LOG_SWITCH:
            logging.info(f"User {username} viewed book with ISBN: {book_isbn}")

        # Simulated book view
        return {
            "isbn": book_isbn,
            "title": "Sample Title",
            "author": "Sample Author"
        }

    @authenticate
    def list_books(username):
        """Allows a user to list all books"""
        if LOG_SWITCH:
            logging.info(f"User {username} requested the book list")

        # Simulated book list
        return [
            {"title": "Book 1", "author": "Author 1", "isbn": "123"},
            {"title": "Book 2", "author": "Author 2", "isbn": "456"},
        ]

    @authenticate
    def reset_password(username):
        """Allows a user to reset their password"""
        new_password = input("Enter new password: ")

        if LOG_SWITCH:
            logging.info(f"User {username} reset their password")

        # Simulated password reset
        return {
            "username": username,
            "new_password": new_password
        }

    # Simulate some actions
    try:
        print(add_book("jdoe"))
        print(update_book("jdoe"))
        print(delete_book("jdoe"))
        print(view_book("jdoe"))
        print(list_books("jdoe"))
        print(reset_password("jdoe"))
    except PermissionError as e:
        print(e)
```

```
2024-07-05 15:47:05 - DEBUG - jdoe is being verified ....
Enter book title: jdoe
Enter book author: john
Enter book ISBN: 1425
2024-07-05 15:48:13 - INFO - User jdoe added book: jdoe, john, 1425
2024-07-05 15:48:13 - DEBUG - jdoe is being verified ....
{'title': 'jdoe', 'author': 'john', 'isbn': '1425'}
Enter book ISBN to update: 1465
Enter new title: my book
Enter new author: priya
2024-07-05 15:48:30 - INFO - User jdoe updated book 1465 to: my book, priya
2024-07-05 15:48:30 - DEBUG - jdoe is being verified ....
{'isbn': '1465', 'new_title': 'my book', 'new_author': 'priya'}
Enter book ISBN to delete: 1465
2024-07-05 15:48:36 - INFO - User jdoe deleted book with ISBN: 1465
2024-07-05 15:48:36 - DEBUG - jdoe is being verified ....
{'isbn': '1465'}
Enter book ISBN to view: 1465
2024-07-05 15:48:43 - INFO - User jdoe viewed book with ISBN: 1465
2024-07-05 15:48:43 - DEBUG - jdoe is being verified ....
2024-07-05 15:48:43 - INFO - User jdoe requested the book list
2024-07-05 15:48:43 - DEBUG - jdoe is being verified ....
{'isbn': '1465', 'title': 'Sample Title', 'author': 'Sample Author'}
[{'title': 'Book 1', 'author': 'Author 1', 'isbn': '123'}, {'title': 'Book 2', 'author': 'Author 2', 'isbn': '456'}]
Enter new password: 8073874785
2024-07-05 15:49:35 - INFO - User jdoe reset their password
{'username': 'jdoe', 'new_password': '8073874785'}
```