1.Task: Build a Library Management System Using Python Modules and Classes

book.py

```python
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
        self.is_borrowed = False

    def mark_borrowed(self):
        self.is_borrowed = True

    def mark_returned(self):
        self.is_borrowed = False

    def __str__(self):
        status = "Borrowed" if self.is_borrowed else "Available"
        return f"Title: {self.title}, Author: {self.author}, Status: {status}"
```

user.py

```python
class User:
    def __init__(self, name, user_id):
        self.name = name
        self.user_id = user_id
        self.borrowed_books = []

    def borrow_book(self, book):
        if len(self.borrowed_books) >= 3:
            print(f"{self.name} cannot borrow more than 3 books.")
            return False
        if book.is_borrowed:
            print(f"'{book.title}' is already borrowed by another user.")
            return False
        book.mark_borrowed()
        self.borrowed_books.append(book)
        print(f"{self.name} borrowed '{book.title}' successfully.")
        return True

    def return_book(self, book):
        if book in self.borrowed_books:
            book.mark_returned()
            self.borrowed_books.remove(book)
            print(f"{self.name} returned '{book.title}' successfully.")
            return True
        else:
            print(f"{self.name} cannot return '{book.title}' as it was not borrowed by them.")
            return False

    def view_borrowed_books(self):
        if self.borrowed_books:
            print(f"Books borrowed by {self.name}:")
            for book in self.borrowed_books:
                print(f"- {book.title} by {book.author}")
        else:
            print(f"{self.name} has not borrowed any books.")

    def __str__(self):
        return f"User: {self.name}, ID: {self.user_id}, Borrowed Books: {len(self.borrowed_books)}"
```

library.py

```
from book import Book
from user import User

class Library:
    def __init__(self):
        self.books = []
        self.users = []

    def add_book(self, title, author):    # Add a new book to the library
        new_book = Book(title, author)
        self.books.append(new_book)

    def remove_book(self, title):      # Remove a book from the library
        for book in self.books:
            if book.title == title and not book.is_borrowed:
                self.books.remove(book)

                return True
        print(f"Book '{title}' not found or is currently borrowed.")
        return False


    def view_books(self):                  # Display all available books
        print("Available books in the library:")
        available_books = [book for book in self.books if not book.is_borrowed]
        if available_books:
            for book in available_books:
                print(book)
        else:
            print("No available books at the moment.")


    def search_books(self, query):                # Search for a book by title or author
        print(f"Search results for '{query}':")
        results = [book for book in self.books if query.lower() in book.title.lower() or query.lower() in
book.author.lower()]
        if results:
            for book in results:
                print(book)
        else:
            print("No matching books found.")

    def register_user(self, name):                          #Register a new user
        user_id = len(self.users) + 1
        new_user = User(name, user_id)
        self.users.append(new_user)
        print(f"User '{name}' registered successfully with User ID: {user_id}.")
        return new_user


    def get_user_by_id(self, user_id):          # Find a user by ID
        for user in self.users:
            if user.user_id == user_id:
                return user
        return None


    def borrow_book(self, user, title):          # Borrow a book
        for book in self.books:
            if book.title == title:
                return user.borrow_book(book)
        print(f"Book '{title}' not found in the library.")
```

```
        return False


    def return_book(self, user, title):    # Return a book
        for book in self.books:
            if book.title == title:
                return user.return_book(book)
        print(f"Book '{title}' not found in the library.")
        return False
```

main.py

```
from library import Library

# Admin password
ADMIN_PASSWORD = "admin123"

def admin_login():
    password = input("Enter admin password: ")
    if password == ADMIN_PASSWORD:
        print("Admin login successful.")
        return True
    else:
        print("Incorrect password. Access denied.")
        return False

def main():
    library = Library()


    library.add_book("The Great Gatsby", "F. Scott Fitzgerald")
    library.add_book("To Kill a Mockingbird", "Harper Lee")
    library.add_book("1984", "George Orwell")

    is_admin_logged_in = False

    while True:
        print("\n===== Library Management System =====")
        print("1. Register User")
        print("2. View Available Books")
        print("3. Search for a Book")
        print("4. Borrow a Book")
        print("5. Return a Book")
        print("6. View Borrowed Books")
        print("7. Admin Login")
        print("8. Add a Book (Admin Only)")
        print("9. Remove a Book (Admin Only)")
        print("10. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            name = input("Enter your name: ")
            user = library.register_user(name)

        elif choice == '2':
            library.view_books()

        elif choice == '3':
            query = input("Enter book title or author to search: ")
            library.search_books(query)
```

```
        elif choice == '4':
            user_id = int(input("Enter your user ID: "))
            user = library.get_user_by_id(user_id)
            if user:
                title = input("Enter the title of the book to borrow: ")
                library.borrow_book(user, title)
            else:
                print("User not found. Please register first.")

        elif choice == '5':
            user_id = int(input("Enter your user ID: "))
            user = library.get_user_by_id(user_id)
            if user:
                title = input("Enter the title of the book to return: ")
                library.return_book(user, title)
            else:
                print("User not found. Please register first.")

        elif choice == '6':
            user_id = int(input("Enter your user ID: "))
            user = library.get_user_by_id(user_id)
            if user:
                user.view_borrowed_books()
            else:
                print("User not found.")

        elif choice == '7':

            is_admin_logged_in = admin_login()

        elif choice == '8':

            if is_admin_logged_in:
                title = input("Enter the title of the book: ")
                author = input("Enter the author of the book: ")
                library.add_book(title, author)
                print(f"Book '{title}' by {author} added to the library.")
            else:
                print("Please login as admin first to add a book.")

        elif choice == '9':

            if is_admin_logged_in:
                title = input("Enter the title of the book to remove: ")
                if library.remove_book(title):
                    print(f"Book '{title}' removed from the library.")
                else:
                    print(f"Book '{title}' could not be removed.")
            else:
                print("Please login as admin first to remove a book.")

        elif choice == '10':
            print("Exiting the system.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Output

```
PS C:\Users\user\Desktop\python\python daily task> & C:/Users/user/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/user/Desktop/python/python daily task/
beinex1510/library/main.py"

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 1
Enter your name: Alice
User 'Alice' registered successfully with User ID: 1.

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 2
Available books in the library:
Title: The Great Gatsby, Author: F. Scott Fitzgerald, Status: Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
Title: 1984, Author: George Orwell, Status: Available
```

```
===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 3
Enter book title or author to search: The Great Gatsby
Search results for 'The Great Gatsby':
Title: The Great Gatsby, Author: F. Scott Fitzgerald, Status: Available

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 4
Enter your user ID: 1
Enter the title of the book to borrow: The Great Gatsby
Alice borrowed 'The Great Gatsby' successfully.
```

```
===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 6
Enter your user ID: 1
Books borrowed by Alice:
- The Great Gatsby by F. Scott Fitzgerald

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 5
Enter your user ID: 1
Enter the title of the book to return: The Great Gatsby
Alice returned 'The Great Gatsby' successfully.
```

```
===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 7
Enter admin password: admin123
Admin login successful.

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 8
Enter the title of the book: dracula
Enter the author of the book: me
Book 'dracula' by me added to the library.
```

```
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 2
Available books in the library:
Title: The Great Gatsby, Author: F. Scott Fitzgerald, Status: Available
Title: To Kill a Mockingbird, Author: Harper Lee, Status: Available
Title: 1984, Author: George Orwell, Status: Available
Title: dracula, Author: me, Status: Available

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 9
Enter the title of the book to remove: dracula
Book 'dracula' removed from the library.

===== Library Management System =====
1. Register User
2. View Available Books
3. Search for a Book
4. Borrow a Book
5. Return a Book
6. View Borrowed Books
7. Admin Login
8. Add a Book (Admin Only)
9. Remove a Book (Admin Only)
10. Exit
Enter your choice: 10
Exiting the system.
PS C:\Users\user\Desktop\python\python daily task> s
```

2. Task: Build a To-Do List Application Using Python Modules and Classes

user.py

```python
class User:
    def __init__(self, name):
        self.name = name
        self.tasks = []

    def add_task(self, task):
        self.tasks.append(task)

    def remove_task(self, task):
        if task in self.tasks:
            self.tasks.remove(task)
        else:
            raise ValueError("Task not found in user's task list.")

    def view_tasks(self):
        if not self.tasks:
            print("No tasks found.")
        else:
            for task in self.tasks:
                print(task)

    def view_pending_tasks(self):
        pending_tasks = [task for task in self.tasks if not task.is_completed]
        if not pending_tasks:
            print("No pending tasks found.")
        else:
```

```
            for task in pending_tasks:
                print(task)

    def view_completed_tasks(self):
        completed_tasks = [task for task in self.tasks if task.is_completed]
        if not completed_tasks:
            print("No completed tasks found.")
        else:
            for task in completed_tasks:
                print(task)
```

task.py

```
from datetime import datetime

class Task:
    def __init__(self, title, description, due_date):
        self.title = title
        self.description = description
        self.due_date = self.validate_due_date(due_date)
        self.is_completed = False

    def validate_due_date(self, due_date):
        if isinstance(due_date, str):
            due_date = datetime.strptime(due_date, "%Y-%m-%d")
        if due_date < datetime.now():
            raise ValueError("Due date cannot be in the past.")
        return due_date

    def mark_completed(self):
        if self.is_completed:
            raise Exception("Task is already completed.")
        self.is_completed = True

    def __str__(self):
        status = "Completed" if self.is_completed else "Pending"
        return f"{self.title} (Due: {self.due_date.date()}, Status: {status})"
```

todolist.py

```
class ToDoList:
    def __init__(self):
        self.users = []

    def add_user(self, user):
        self.users.append(user)

    def remove_user(self, user):
        if user in self.users:
            self.users.remove(user)
        else:
            raise ValueError("User not found in the system.")

    def get_user_tasks(self, user):
        if user in self.users:
            return user.tasks
        else:
            raise ValueError("User not found in the system.")
```

main.py

```
from task import Task
from user import User
from todolist import ToDoList

def main():
    todo_list = ToDoList()

    while True:
        print("\n1. Create User\n2. Add Task\n3. View Tasks\n4. Complete Task\n5. Remove Task\n6. Exit")
        choice = input("Choose an option: ")

        if choice == '1':
            name = input("Enter user name: ")
            user = User(name)
            todo_list.add_user(user)
            print(f"User {name} created.")

        elif choice == '2':
            user_name = input("Enter user name: ")
            user = next((u for u in todo_list.users if u.name == user_name), None)
            if user:
                title = input("Task title: ")
                description = input("Task description: ")
                due_date = input("Due date (YYYY-MM-DD): ")
                try:
                    task = Task(title, description, due_date)
                    user.add_task(task)
                    print("Task added.")
                except ValueError as e:
                    print(e)

            else:
                print("User not found.")

        elif choice == '3':
            user_name = input("Enter user name: ")
            user = next((u for u in todo_list.users if u.name == user_name), None)
            if user:
                user.view_tasks()
            else:
                print("User not found.")

        elif choice == '4':
            user_name = input("Enter user name: ")
            task_title = input("Enter task title to mark as completed: ")
            user = next((u for u in todo_list.users if u.name == user_name), None)
            if user:
                task = next((t for t in user.tasks if t.title == task_title), None)
                if task:
                    try:
                        task.mark_completed()
                        print(f"Task '{task_title}' marked as completed.")
                    except Exception as e:
                        print(e)
                else:
                    print("Task not found.")
            else:
                print("User not found.")

        elif choice == '5':
            user_name = input("Enter user name: ")
            task_title = input("Enter task title to remove: ")
```

```
            user = next((u for u in todo_list.users if u.name == user_name), None)
            if user:
                task = next((t for t in user.tasks if t.title == task_title), None)
                try:
                    user.remove_task(task)
                    print(f"Task '{task_title}' removed.")
                except ValueError as e:
                    print(e)
            else:
                print("User not found.")

        elif choice == '6':
            print("Exiting application.")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Output

```
1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 1
Enter user name: John
User John created.

1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 2
Enter user name: John
Task title: Save nature
Task description: xyz
Due date (YYYY-MM-DD): 2024-11-01
Task added.

1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 3
Enter user name: John
Save nature (Due: 2024-11-01, Status: Pending)
```

```
1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 4
Enter user name: John
Enter task title to mark as completed: Save nature
Task 'Save nature' marked as completed.

1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 3
Enter user name: John
Save nature (Due: 2024-11-01, Status: Completed)

1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 5
Enter user name: John
Enter task title to remove: Save nature
Task 'Save nature' removed.
```

```
1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 3
Enter user name: John
No tasks found.

1. Create User
2. Add Task
3. View Tasks
4. Complete Task
5. Remove Task
6. Exit
Choose an option: 6
Exiting application.
PS C:\Users\user\Desktop\python\python daily task>
```