

Library Space Management System

Huda Mukhtar

Calvin Dong

Elaine Liu

Connor Haines

Problem Definition

We need a booking system to manage study space bookings to help increase productivity and save time looking for study space.

We need a booking system that allows users to filter for rooms with various resources (e.g. seating, accessibility needs, outlets availability) to fit their needs.

We need a booking system that has an admin mode to give librarians special permissions to make special room bookings, add, and remove spaces that are available.

Project Objective

The objective of this system is to allow students to filter for and book study spaces. It should also allow librarians to add and delete study spaces to the system, as well as override booking limits.

Functional Requirements

- A user should be able to visually see the available tables and timeslots
- A user should be able to book a study space
- A user should be able to cancel a study space
- The user should be able to view the resources of an available study space (resources being outlet availability, seating, media, accessibility)
- A user should be able to specify filters and see available spaces for the current and next six days
- The librarian should be able to add a table for students to use
- The librarian should be able to remove a table for use

Non-Functional Requirements

- Bookings should only be available during library open hours of the next 7 days
- Students can't book more than 2 hours per day
- Librarians can book more than 2 hours per day
- Students and Librarians must login with a UWO email to verify their status

Stakeholders List

The stakeholders of this application include **library staff** and **students who use the library**. Librarians would be interested in using the application to list the spaces available to study for students. Students would be interested in this application as they would be saving time by having a one-stop spot for finding study spaces that are available for them.

Success / Acceptance Criteria for each Stakeholder

Student

- **As a student, I want to see the available study spaces.**
 - Given the student signs in using their UWO email, then the student should be able to view the spaces available for the next 7 days.
 - Given the student applies filters to find study spaces with specific constraints, then only the available study spaces with those resources should be displayed.
 - Given the student has applied filters and the applicable spaces have been listed, the resources available for the study spaces should be displayed.
- **As a student, I want to be able to sign up for a study space.**
 - Given the student has signed in using their UWO email, and selects a time slot to book, and meets the prerequisites to book the time slot (less than 2 hours, time slot is not already booked), then the time slot will be reserved for the student.
- **As a student, I want to be able to view my bookings**
 - Given the student has bookings for future dates, the student should be able to view the bookings they previously made.
- **As a student, I want to be able to cancel my study space**
 - Given the student has booked a time slot, they should be able to see their time slot and cancel their booking

Librarian

The librarian has the same acceptance criteria as a Student as well as the following additional acceptance criteria:

- **As a librarian, I should be able to book a timeslot for longer than 2 hours**
 - Given the librarian has signed in using their UWO email, they will not receive error messages when booking time slots for more than 3 hours.
- **As a librarian, I should be able to add a space into the system with custom attributes**
 - Given the librarian has signed in using their UWO email, they will be given an option to add new tables with certain attributes such as accessibility, number of seats, and location.
- **As a librarian, I should be able to remove a space from the system.**
 - Given the librarian has signed in using their UWO email, they should be able to view all existing spaces.
 - Given the librarian selects the option to remove an existing space and inputs the spaceId to remove, then the space should be removed from the database and should no longer be viewable by students.
 - Given the librarian selects to remove the space, then all bookings for the removed space should also be removed.

Use Case Diagram

The use case diagram showcases the numerous interactions that the actors have with the system. The actor Student may Login, View Study Space, Book a study space, View booking, and Cancel a Booking. While the Librarian Actor can perform the same functions as a Student Actor, with the addition of Add Study Space and Remove Study Space. All actors must be authenticated through the Login function prior to performing any of the above listed actions.

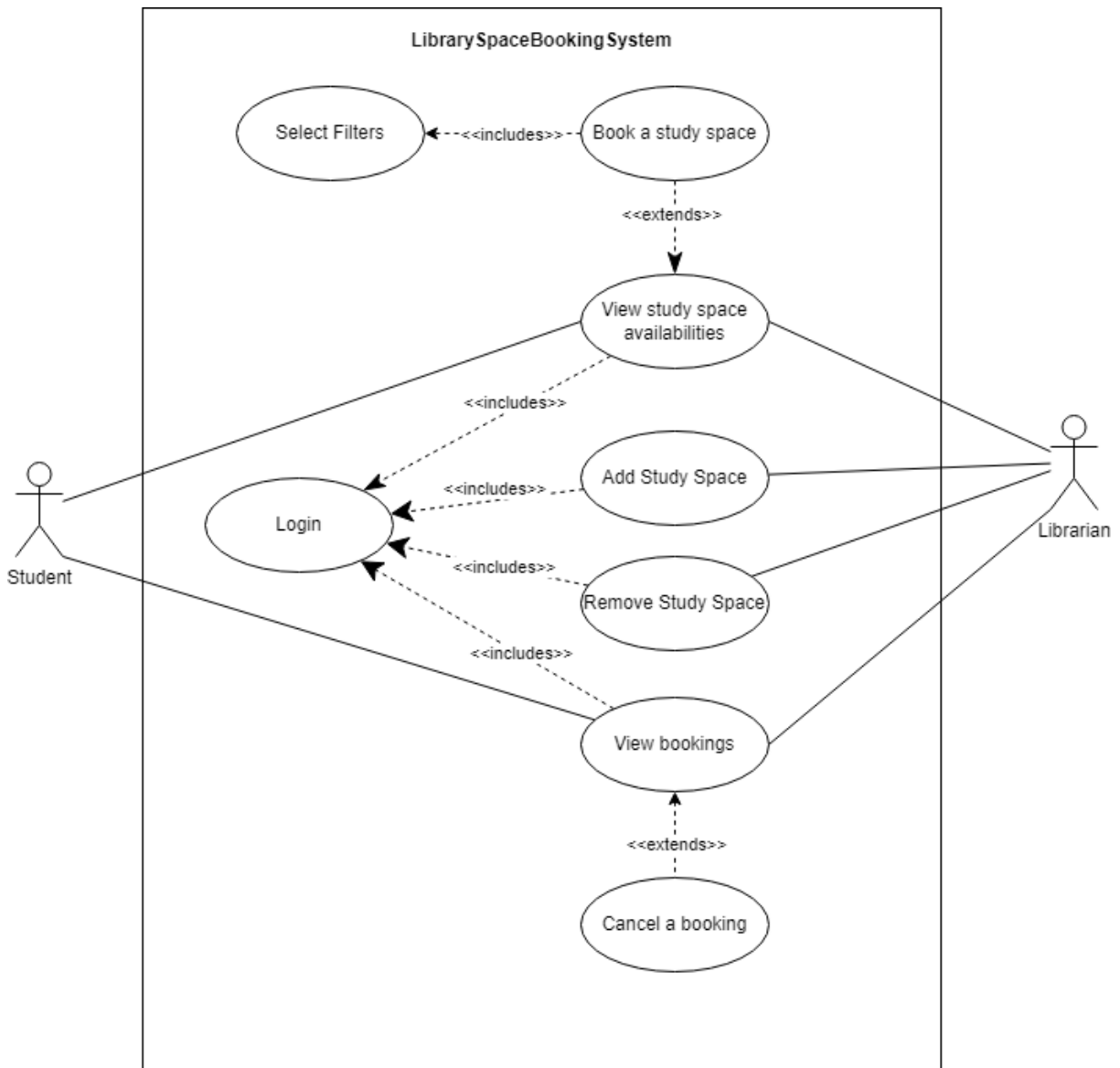


Figure 1 - Use Case Diagram

Use Case Descriptions

Use Case Name:	Booking a study space as a student					
Scenario:	A student books a timeslot for a study space, up to a maximum of 2 hours					
Triggering Event:	Student logs into the library booking management application. Student views available study spaces.					
Brief Description	When a student opens up the booking management application, the Library Space Management System verifies student information using their UWO ID and password. The student selects the option to view available spaces and has the option to make a booking for a specific time and location.					
Actor:	Student					
Related Use Case:	Includes: Verify status by UWO ID Extends: View study space availabilities					
Stakeholders	Librarian: to maintain a list of available study spaces					
Preconditions:	User must exist. User must be authenticated. Study space, available tables, and their time slots must exist to make a booking.					
Postconditions	Booking must be created for the student. The chosen time slot must become unavailable to other student bookings. Timeslot must be associated with that student.					
Flow of Activities	<table><tr><th>Actor</th><th>System</th></tr><tr><td><div>1. Student log into management system</div><div>2. Student selects prompt to view booking slots</div><div>3. Student selects option to book a timeslot</div><div>4. Student answer prompts for study space filters and preferences about the booking space</div><div>5. Student chooses their time slot and confirms booking</div></td><td><div>1.1 System verifies student login credentials</div><div>2.1 System displays all bookings as a table</div><div>4.1 System displays options that fit students needs</div><div>5.1 System updates students current booking and update the database</div></td></tr></table>		Actor	System	<div>1. Student log into management system</div> <div>2. Student selects prompt to view booking slots</div> <div>3. Student selects option to book a timeslot</div> <div>4. Student answer prompts for study space filters and preferences about the booking space</div> <div>5. Student chooses their time slot and confirms booking</div>	<div>1.1 System verifies student login credentials</div> <div>2.1 System displays all bookings as a table</div> <div>4.1 System displays options that fit students needs</div> <div>5.1 System updates students current booking and update the database</div>
Actor	System					
<div>1. Student log into management system</div> <div>2. Student selects prompt to view booking slots</div> <div>3. Student selects option to book a timeslot</div> <div>4. Student answer prompts for study space filters and preferences about the booking space</div> <div>5. Student chooses their time slot and confirms booking</div>	<div>1.1 System verifies student login credentials</div> <div>2.1 System displays all bookings as a table</div> <div>4.1 System displays options that fit students needs</div> <div>5.1 System updates students current booking and update the database</div>					
Exception Conditions	1.1 If a student enters wrong credentials or does not exist, the system will prompt them to login again. 4.1 May not have spaces that fit the students' needs. If so, the system will notify the student					

	<p>that no spaces exist.</p> <p>5.1 If a student selects a time slot that already has a booking for that study space, they will receive an error message and return to the main menu.</p>
--	---

Use Case Name:	Adding a study space as a librarian					
Scenario:	A librarian adds a new study space to the library management system					
Triggering Event:	A user that has been verified as a librarian will be able to add a study space through the management application.					
Brief Description	If a new study space has been opened in the library, the librarian should be able to add this space to be booked. Features like name, number of seats, accessibility, outlets, quiet zone, and media will be characteristics that the librarian can enter into the system					
Actor:	Librarian					
Related Use Case:	Includes: Verify status by UWO ID					
Stakeholders	Students: are the ones who can book and use these new study spaces					
Preconditions:	User must exist. User has to be authenticated as a librarian.					
Postconditions	Study space must be created after using the characteristics described. Users can book this newly created study space. Study space will be available to be seen by all users. Librarian can remove the added space.					
Flow of Activities	<table><tr><th>Actor</th><th>System</th></tr><tr><td>1. Librarian log into management system 2. Librarian selects prompt to add study space 3. Librarian enter characteristics about the study space</td><td>1.1 User is authenticated as a librarian 3.1 Study space gets added to the JSON database and updated within the system</td></tr></table>		Actor	System	1. Librarian log into management system 2. Librarian selects prompt to add study space 3. Librarian enter characteristics about the study space	1.1 User is authenticated as a librarian 3.1 Study space gets added to the JSON database and updated within the system
Actor	System					
1. Librarian log into management system 2. Librarian selects prompt to add study space 3. Librarian enter characteristics about the study space	1.1 User is authenticated as a librarian 3.1 Study space gets added to the JSON database and updated within the system					
Exception Conditions	1.1 If librarian enters wrong credentials or does not exist, the system will prompt them again to login 1.2 If user is not set as a librarian, they will not have the options to add a study space					

Sequence Diagrams

1) Booking a study space as a student

Users must first login to the system and be validated. Upon validation, the session is created by referring to the database and loading the most recent information. The user is brought back to the main menu page where they can select to view study spaces. Study spaces for the next 7 days are retrieved by the session and displayed for the User. The user can select their desired booking date and answer preferred filters. The session will find available spaces that fit the filters, then prompt the user for a time and duration. If successful, the booking will be updated in the database, if failed, the session will notify the user.

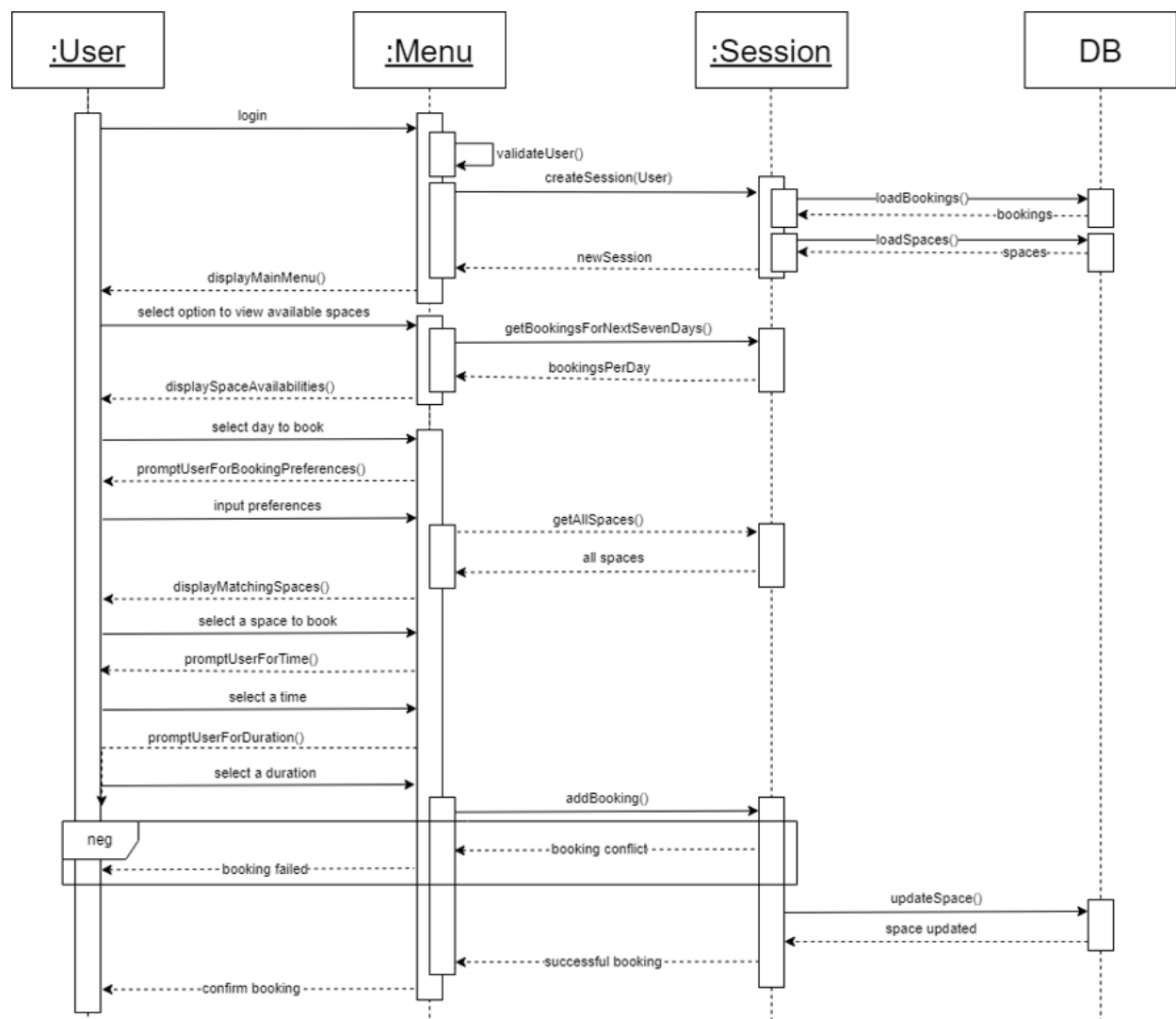


Figure 2 - Sequence Diagram #1

2) Adding a study space as a librarian

Librarians must first login to the system and be validated. Upon validation, the session is created by referring to the database and loading the most recent information. The Librarian is brought back to the main menu page and prompted with a series of questions. Once all questions are answered, the system will update the session and database with the new study space. The session will confirm to the user that the space has been added.

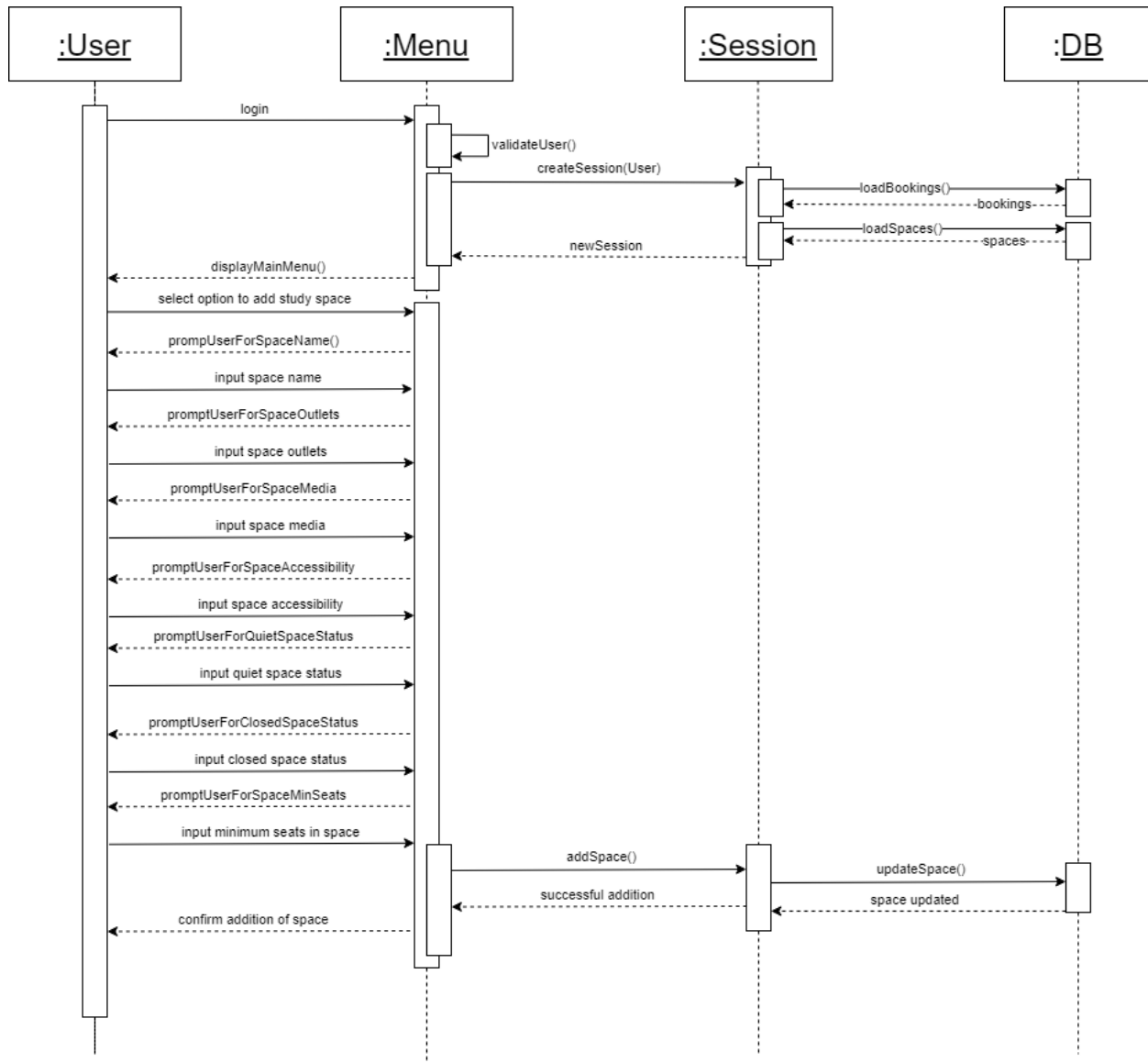


Figure 3 - Sequence Diagram #2

System Architecture

Our system is a monolithic application that incorporates the Layered Architecture Design Pattern. There are 3 layers: the Presentation layer, the Business layer, and the Database layer. The Presentation layer consists of the front end where the user will interact and answer prompts from. Once user input is received, the Business layer will apply logic to the prompts and read/write data to the Database Layer which contains the JSON files for the Spaces, Users, and Bookings.

Layered Architecture

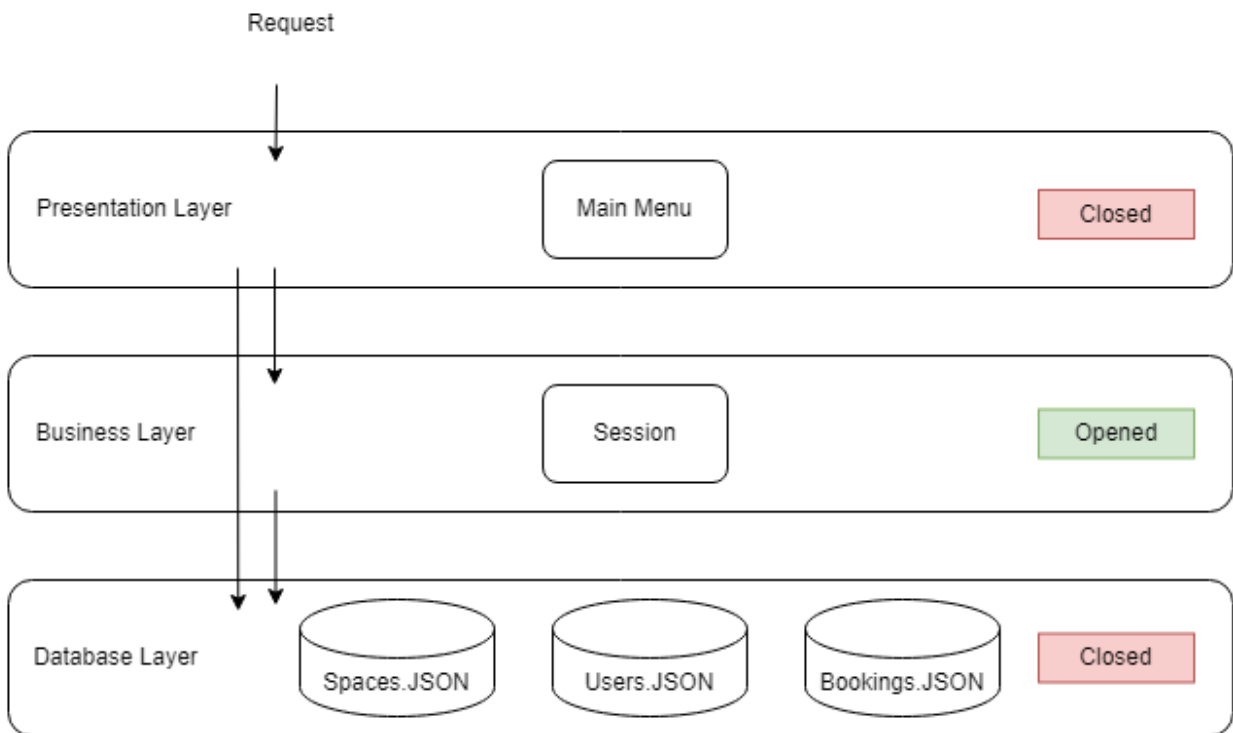


Figure 4 - System Architecture

Detailed Class Diagram

The below Detailed Class Diagram represents the Library Space Management System.

The main driver of the system is the *Menu* class, which acts as the primary user interface and UI. This class will deal with the user that logs in, validate their identity, and provide a menu with the list of actions that a user can take regarding Library Spaces.

The *Session* class encompasses the business logic for the system and is created by the Menu class. The Session class has 1 User who is signed in, a list of bookings, and a list of the study spaces which are extracted from our json files. Its functions deal with making changes to study spaces and bookings based on the user's status and selection from the menu options.

The *User* class is generated by the session to keep track of the user's status as to being a librarian, bookings, and userId for when they decide to make a booking.

The *Booking* class is generated by a *Session* when the user decides to make a booking. This would require the user to select a space, date, and duration of booking. The booking would then be added to the database with its bookingId.

The *StudySpace* class is generated by a *Session* when a user decides to view a space. This contains the details of the space and its accessibility features.

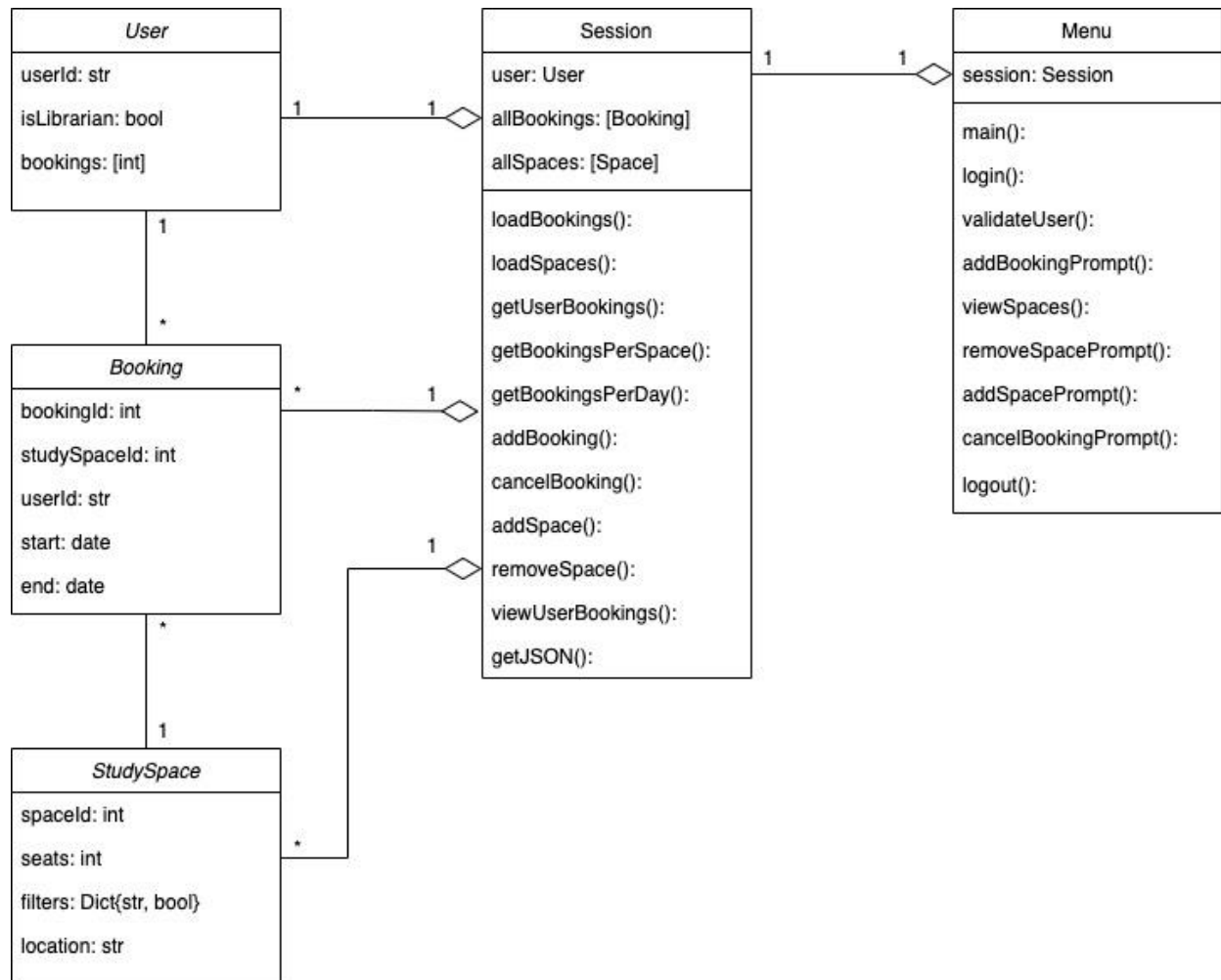


Figure 5 - Class Diagram

State-machine diagram

The state diagram consists of different states of the Library Space Management System depending on the type of user. As the application is opened, it is in a state of waiting for the user to log in. Depending on which user logs in, a different state is reached. In the *Logged in as student* state, there are multiple states that can be reached: *Booking Displayed*, *Booking Canceled*, *Available Spaces Displayed* and *Booking confirmed*. The *Logging in as Librarian* state contains the same states, but with additional states: *New study space added* and *Study space deleted* state. With each of these options, the user can return to go back to the main menu and logout.

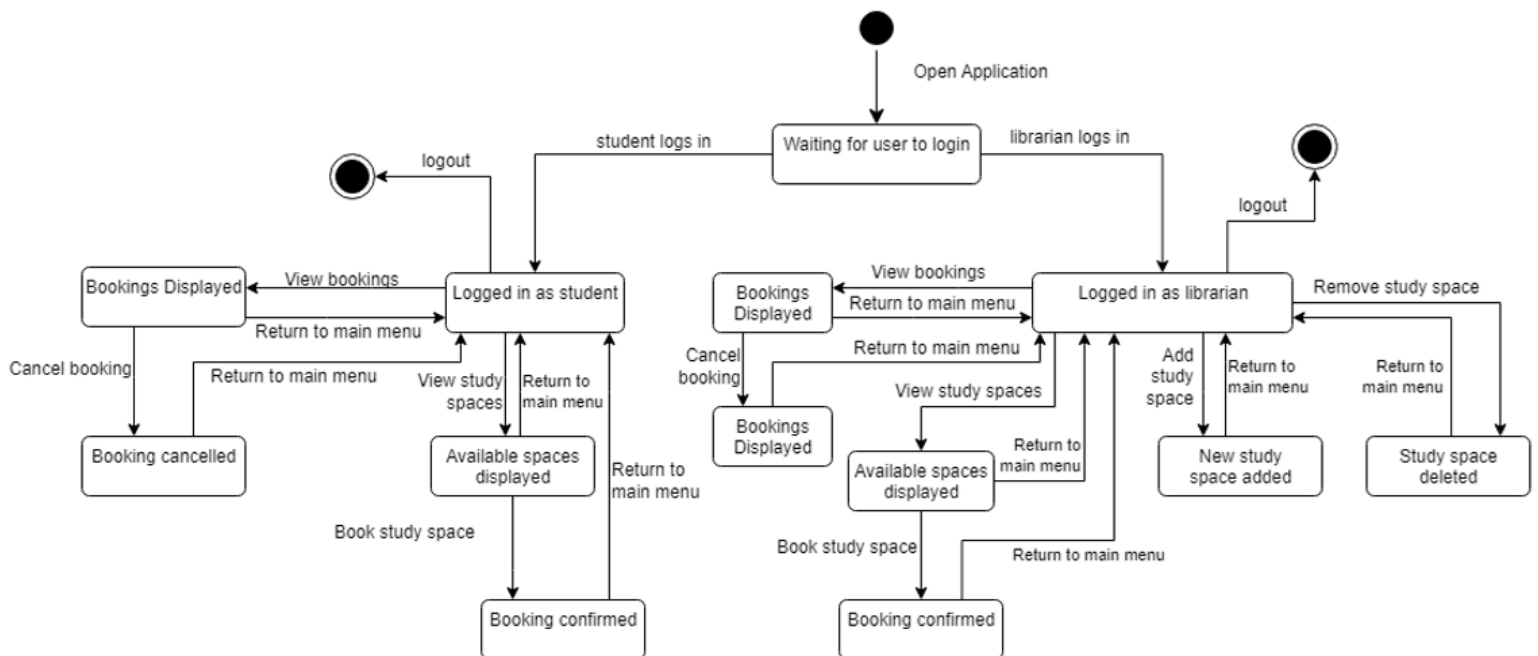


Figure 6 - State Machine Diagram (Entire System)

ER Diagram (Data modeling)

A User can either be a Librarian or Student. A User can create one or many bookings which have a Study Space. Every study space contains resources which act as filters and characteristics about the study space. Bookings contain a unique BookingId as well as the start/end time. Librarians have the ability to Add/Remove a study space, if a StudySpace is removed with an associated Booking, the booking will be removed as well.

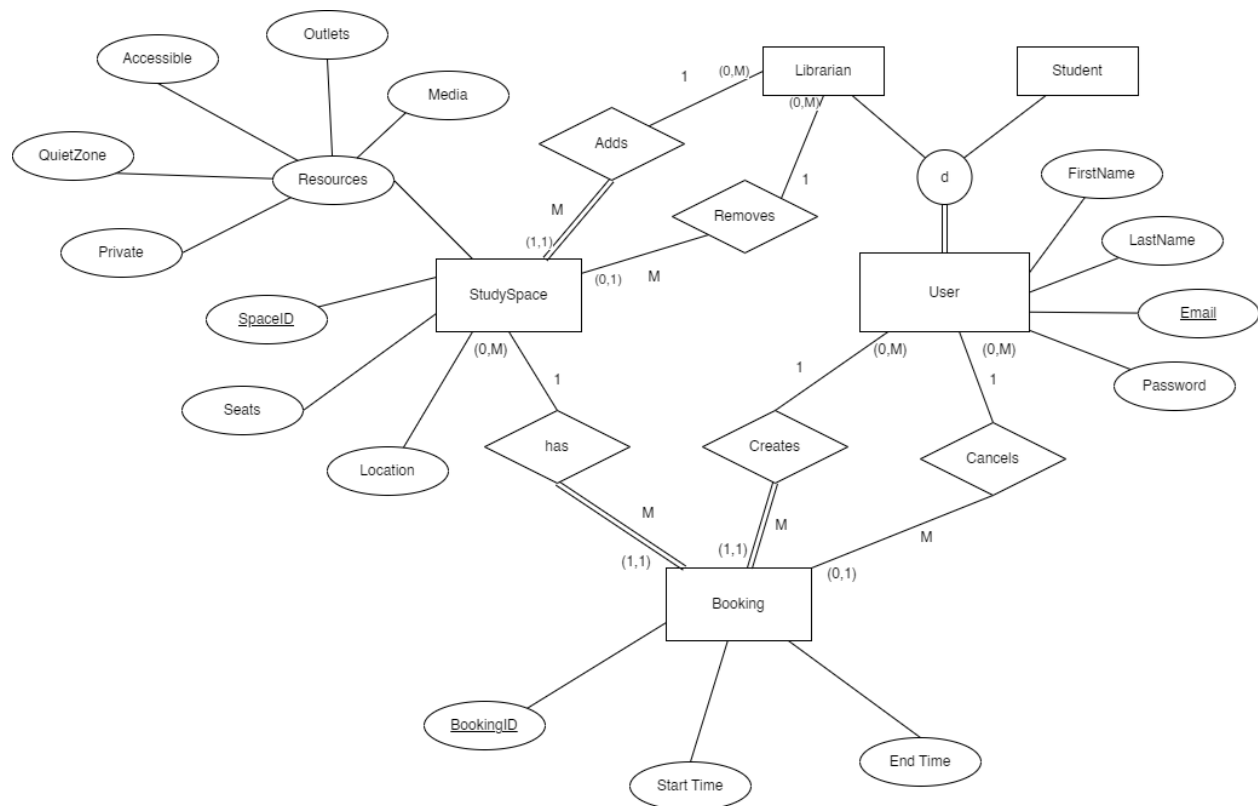


Figure 7 - ER Diagram

Source Code

GitHub link: <https://github.com/connor2033/LibrarySpaceBookingSystem>

Conclusions

Through this project, we took away important lessons about project management and the software development lifecycle. Key takeaways from this project include the importance of organization, communication, planning, and project definition and scoping. Firstly, to stay organized, we created checklists and deadlines so that we were able to keep tasks on track and know who's accountable for what. This allowed us to split up work so we could finish tasks faster. Secondly, having open communication helped troubleshoot problems more efficiently and quickly. Since everyone had their own skills and familiarities, having open communication allowed us to work through blockers faster. Thirdly, understanding the problem definition, objectives, and stakeholders helped us figure out features we should have in our system. This helped us with our planning stages and designing the functions our application should have. Finally, thorough planning by defining diagrams, use cases, and the system architecture allowed us to have a clear understanding on how the system should look like. This made it easier to reference back to the diagrams to know what needs to be built and how they will be built.

References

Docker documentation. Docker Documentation. (2022, November 30). Retrieved November 30, 2022, from <https://docs.docker.com/>

Full PYTEST documentation. Full pytest documentation - pytest documentation. (n.d.). Retrieved November 30, 2022, from <https://docs.pytest.org/en/7.2.x/contents.html>

Introduction - Rich 12.6.0 documentation. (n.d.). Retrieved November 30, 2022, from <https://rich.readthedocs.io/en/stable/introduction.html>

Appendix A. Project WBS

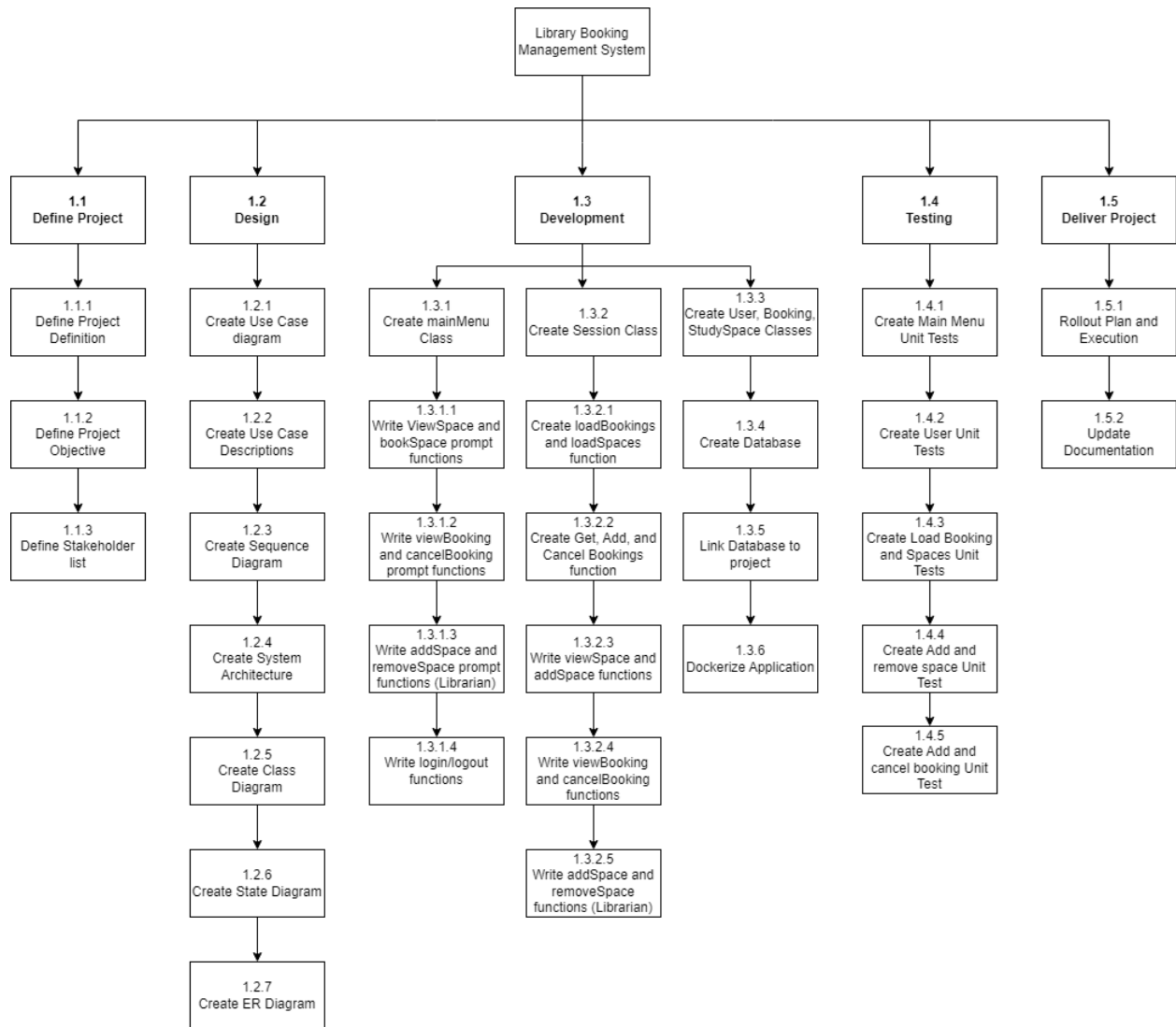


Figure 8 - WBS Structure Diagram

Appendix B. Task Assignment Matrix

WBS ID	Tasks	Task Owner	Support
1.1.1	Define Project Definition	Huda Mukhtar Calvin Dong Elaine Liu Connor Haines	
1.1.2	Define Project Objective	Huda Mukhtar Calvin Dong Elaine Liu Connor Haines	
1.1.3	Define Stakeholder List	Huda Mukhtar Calvin Dong Elaine Liu Connor Haines	
1.2.1	Create Use Case Diagram	Huda Mukhtar	
1.2.2	Create Use Case Descriptions	Huda Mukhtar	Calvin Dong
1.2.3	Create Sequence Diagrams	Huda Mukhtar Connor Haines Elaine Liu	Calvin Dong
1.2.4	Create System Architecture	Elaine Liu Connor Haines	Huda Mukhtar Calvin Dong
1.2.5	Create Class Diagram	Connor Haines Huda Mukhtar Calvin Dong Elaine Liu	
1.2.6	Create State Diagram	Elaine Liu	Calvin Dong Connor Haines
1.2.7	Create ER Diagram	Elaine Liu	
1.3.1	Create MainMenu Class	Huda Mukhtar Connor Haines	Calvin Dong Elaine Liu
1.3.1.1	Write ViewSpace and bookSpace prompt functions	Huda Mukhtar	Connor Haines
1.3.1.2	Write viewBooking and cancelBooking prompt functions	Elaine Liu	

1.3.1.3	Write addSpace and removeSpace prompt functions (Librarian)	Calvin Dong	
1.3.1.4	Write login/logout functions	Connor Haines	
1.3.2	Create Session Class	Elaine Liu	Connor Haines
1.3.2.1	Create loadBookings and loadSpaces function	Elaine Liu	
1.3.2.2	Create Get, Add, and Cancel Bookings function	Elaine Liu	Connor Haines
1.3.2.3	Write viewSpace and addSpace functions	Calvin Dong	
1.3.2.4	Write viewBooking and cancelBooking functions	Elaine Liu	Connor Haines
1.3.2.5	Write addSpace and removeSpace functions (Librarian)	Calvin Dong	
1.3.3	Create User, Booking, StudySpace Classes	Connor Haines	Elaine Liu
1.3.4	Create Database	Connor Haines	
1.3.5	Link Database to project	Connor Haines	
1.3.6	Dockerize Application	Connor Haines Elaine Liu	Huda Mukhtar
1.4.1	Create Login Unit Tests	Elaine Liu	
1.4.3	Create Booking Unit Tests	Elaine Liu	
1.4.4	Create StudySpace Unit Tests	Elaine Liu	
1.5.1	Rollout Plan and Execution	Huda Mukhtar Calvin Dong Elaine Liu Connor Haines	
1.5.2	Update Documentation	Huda Mukhtar Calvin Dong Elaine Liu Connor Haines	

Appendix C. Sample of Commits











Merge pull request #34 from connor2033/final ... connor2033 committed 19 hours ago	Verified		1b48c5e	<>
Fixed duration bug, updated readme connor2033 committed 19 hours ago			c7e95f2	<>
bug fixes hudamukhtar1 committed 21 hours ago			f499e24	<>
Merge pull request #33 from connor2033/fix-bug ... eliu72 committed yesterday	Verified		2aa9086	<>
Add a check to make sure user is booking a time in the future eliu72 committed yesterday			f07b36c	<>
Fixed screen clearing issue for multiple OS connor2033 committed yesterday			f2edff1	<>
Merge pull request #31 from connor2033/refactor ... connor2033 committed yesterday	Verified		104c5ad	<>
Additional refactoring and cleanup connor2033 committed yesterday			df5554c	<>
Merge pull request #30 from connor2033/fix-booking-overlap ... eliu72 committed yesterday	Verified		bb355d5	<>
Refactor modules and fix test cases eliu72 committed yesterday			74ef61a	<>

Figure 9 - Sample of Commits

Appendix D. Test Cases

Test Case	Module	Objective	Prerequisite	Steps	Test Data	Expected result	Actual Result
test_loadBookings	session.py	Verify functionality of loadBookings()	1. Provide mock data 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Call loadBookings() and compare with expected result	List of booking dictionaries	Dictionary of Booking objects	Verified
test_loadSpaces	session.py	Verify functionality of loadSpaces()	1. Provide mock data 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Call loadSpaces() and compare with expected result	List of space dictionaries	Dictionary of Space objects	Verified
test_addBooking	session.py	Verify functionality of addBooking()	1. Provide mock data 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Add a booking and verify it was added to system and database	New booking details	New booking created in system and in database	Verified
test_cancelBooking	session.py	Verify functionality of cancelBooking()	1. Add a space 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Cancel newly added booking and verify it was removed from system and database	Booking details	Booking is removed from system and from database	Verified

test_addSpace	session.py	Verify functionality of addSpace()	1. Provide mock data 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Add a space and verify it was added to system and database	New space details	New space created in system and in database	Verified
test_removeSpace	session.py	Verify functionality of removeSpace()	1. Add a space 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Cancel newly added space and verify it was removed from system and database	Space details	Space is removed from system and from database	Verified
test_validateUser	login.py	Verify functionality of validateUser()	1. Provide mock data 2. Patch file opening methods	1. Mock file opening calls (we don't want to actually open files) 2. Validate a user and check that correct credentials returns a valid user 3. Validate a user and check that incorrect credentials (wrong email or password) returns an invalid user	User details	User is verified accurately	Verified