Hotel Reservation System

Catherine McIlroy, Student No 23173190

Table of Contents

Agile Methodology (Task 9)

For the development phase of the Hotel Reservation System described in Part 1, the project team will follow the scrum framework.

Agile scrum methodology involves breaking the development process down into smaller chunks called “sprints”. Each sprint is typically 2-4 weeks in length, and will focus on development of a limited number of software features with the goal of delivering a working product at the end of each sprint. One sprint can be described as one iteration of the process. For the purposes of this project, each sprint will be two weeks in length.

Before the project begins, there will be a meeting where the goals of the project will be defined and the product backlog is created. The product backlog is a list of all work to be carried out on the project as a whole, and typically consists of user stories with corresponding story points. A user story is a description of a software feature which is written from the perspective of the user, and explains how this feature will provide value for the end user of the software. It should be simple enough that it can be delivered in a single sprint. The corresponding story points represent the effort required for the team to implement these changes, and the scale used can vary. For the purposes of this project we will apply a scale of 1-10, with 1 being the lowest and 10 being the highest. The product backlog is listed in priority order from highest to lowest, and can be modified between sprints if necessary.

Prior to the start of each sprint there will then be a sprint planning meeting, where features to be worked on during a sprint are selected from the product backlog and added to the sprint backlog. This forms a list of user stories to be completed during the current sprint. Team members will choose tasks from the sprint backlog to work on during the current cycle. The sprint backlog is continuously updated as tasks are completed, and can be modified by any scrum team member at any time during the sprint if necessary.

During each sprint, there will be no interference from outside of the immediate scrum team, which will allow team members to focus on the tasks at hand without distractions. There will be a daily scrum meeting where the team can discuss their progress. The scrum sprint is completed once all tasks from the sprint backlog are cleared, and a review of the sprint can then take place.

The agile scrum process differs from the traditional waterfall process in several key areas, as described below.

- **Agile Scrum**
  - Iterative approach, product development occurs in cycles and different stages may occur concurrently
  - Fast delivery of product (2-4 weeks)
  - Produces a working, shippable product increment with every sprint
  - Allows for continuous change to the product and input from the client, allowing modifications to be made to the next increment
  - Client can provide feedback after each product increment
  - Project goals or description may be modified at any time during the development process

- **Waterfall Method**
  - Sequential process, development is divided into distinct stages which occur consecutively
  - Product delivered at the end of the process (up to 2 years)
  - No working product is released until the end of the cycle
  - Client can provide feedback only once the end product is released
  - Does not allow for changes to be made to the product requirements once the development process has started

Agile Artifacts (Task 10)

Shown below are the user stories for this project. Each user story displays a variety of attributes, including its status (to do, in progress or completed), its priority, the team member it has been assigned to, time spent so far, estimated time remaining and how this compares to the original time estimate. It also shows whether this user story has been assigned to a sprint cycle, and gives a story points estimate.

[SCRUM-1] As a system user, I want to be able to log in to the reservation system, so that I can use the features of the system. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Highest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|hzzzzz:
Sprint:	SCRUM Sprint 1
Story point estimate:	5

Description

Add log in feature.

[SCRUM-2] As a system user, I want to be able to log out of the reservation system, so that my account is private and secure. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Highest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i00007:
Sprint:	SCRUM Sprint 1
Story point estimate:	5

[SCRUM-3] As a customer, I want to make a new reservation, so that I can stay at the hotel. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Highest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0000f:
Sprint:	SCRUM Sprint 1
Story point estimate:	7

Description

Allow customer to make a new booking. Involves first checking room availability for that date, so room availability checking feature must be added first.

[SCRUM-4] As a customer with an existing reservation, I want to modify my reservation, so that it meets the needs of my changed plans. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0000n:
Sprint:	
Story point estimate:	7

Description

Add feature to allow changes to dates of reservation and/or number of rooms.

[SCRUM-5] As a customer with an existing reservation, I want to cancel my reservation, so that it meets the needs of my changed plans.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0000v:
Sprint:	
Story point estimate:	5

Description

Add feature to allow customer to cancel an existing reservation.

[SCRUM-6] As a customer considering making a reservation, I want to check current room availability, so that I can see if there is availability for my chosen date(s).

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Highest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0000b:
Sprint:	SCRUM Sprint 1
Story point estimate:	6

Description

Allow customer to enter start and end dates of stay, and display available rooms for that time period, inclusive of start and end dates.

[SCRUM-7] As a customer with a newly made reservation, I want to pay for that reservation, so that there is no outstanding bill. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Medium
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i00017:
Sprint:	
Story point estimate:	8

Description

Add secure payment gateway for online payments. Feature should include method to calculate total payment due for customer's reservation.

[SCRUM-8] As a customer with past bookings, I want to view my reservation history, so that I can see the details of my previous reservations. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Low
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0001j:
Sprint:	
Story point estimate:	7

Description

Generate list of previous reservations including dates of stay, number of rooms, type of room, number of guests, total cost and any other relevant details.

[SCRUM-9] As a hotel receptionist, I want to confirm a newly made reservation, so that the customer receives booking confirmation.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Highest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0000h:
Sprint:	SCRUM Sprint 1
Story point estimate:	2

Description

Add feature to confirm a newly made reservation, and automate sending an email confirmation to the customer who made the reservation.

[SCRUM-10] As a hotel receptionist, I want to check guests in or out of the hotel, so that there is a current record of all guests staying in the hotel.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Medium
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0001z:
Sprint:	
Story point estimate:	5

Description

Add feature to system to allow receptionist to check in a guest when they arrive, and check them out when they leave.

[SCRUM-11] As a hotel receptionist, I want to confirm payment made by a customer, so that the customer will receive a confirmation email.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Medium
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i00027:
Sprint:	
Story point estimate:	2

Description

Add feature to allow receptionist to confirm payment made by a customer. Automate sending email confirmation of payment to the customer.

[SCRUM-12] As a hotel receptionist, I want to make a new reservation on behalf of a customer, so that the customer can stay at the hotel.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0002f:
Sprint:	
Story point estimate:	7

Description

Add feature to receptionist profile to allow receptionist to make a new reservation on behalf of a customer. Feature will be similar to that of https://catmcilroy.atlassian.net/browse/SCRUM-3

[SCRUM-13] As a hotel receptionist, I want to modify an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0002n:
Sprint:	
Story point estimate:	8

Description

Add feature to allow receptionist profile type to access a customer’s existing booking and make changes to it.

[SCRUM-14] As a hotel receptionist, I want to cancel an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0002v:
Sprint:	
Story point estimate:	8

Description

Add feature to allow receptionist profile type to access a customer’s existing booking and cancel it.

[SCRUM-15] As a hotel receptionist, I want to check room availability, so that I can advise a customer whether there is availability for their chosen date(s).

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	High
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i00033:
Sprint:	
Story point estimate:	6

Description

Add feature to allow receptionist profile type to enter dates of stay and number of rooms required. Display list of rooms with corresponding availability. Similar to https://catmcilroy.atlassian.net/browse/SCRUM-6

[SCRUM-16] As a hotel manager, I want to add or remove rooms from the room inventory, so that the room inventory is kept up to date.

Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Low
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0003b:
Sprint:	
Story point estimate:	5

Description

Allow hotel manager profile type to edit room inventory.

[SCRUM-17] As a hotel manager, I want to generate periodical reports, so that I can identify trends in reservations made. Created: 22/Apr/24 Updated: 22/Apr/24

Status:	To Do
Project:	Hotel Reservation System
Components:	None
Affects versions:	None
Fix versions:	None

Type:	Story	**Priority:**	Lowest
Reporter:	Catherine McIlroy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original estimate:	Not Specified		

Rank:	0|i0003j:
Sprint:	
Story point estimate:	4

Description

Allow hotel manager type to generate reports detailing number of reservations made on each date.

Included below is the product backlog prior to the first sprint. This consists of a list of all the work to be carried out on the project (all of the current user stories). It is ordered in descending order of priority level from highest priority tasks (⏫) to lowest priority tasks (⏬).

T	Summary	Assignee	Reporter	P	Status	Resolution	Story point estimate
	As a system user, I want to be able to log in to the reservation system, so that I can use the features of the system.	*Unassigned*	Catherine McIlroy	⏫	TO DO	*Unresolved*	5
	As a system user, I want to be able to log out of the reservation system, so that my account is private and secure.	*Unassigned*	Catherine McIlroy	⏫	TO DO	*Unresolved*	5
	As a customer, I want to make a new reservation, so that I can stay at the hotel.	*Unassigned*	Catherine McIlroy	⏫	TO DO	*Unresolved*	7
	As a customer considering making a reservation, I want to check current room availability, so that I can see if there is availability for my chosen date(s).	*Unassigned*	Catherine McIlroy	⏫	TO DO	*Unresolved*	6
	As a hotel receptionist, I want to confirm a newly made reservation, so that the customer receives booking confirmation.	*Unassigned*	Catherine McIlroy	⏫	TO DO	*Unresolved*	2
	As a customer with an existing reservation, I want to modify my reservation, so that it meets the needs of my changed plans.	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	7
	As a customer with an existing reservation, I want to cancel my reservation, so that it meets the needs of my changed plans.	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	5
	As a hotel receptionist, I want to make a new reservation on behalf of a customer, so that the customer can stay at the hotel.	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	7
	As a hotel receptionist, I want to modify an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	8
	As a hotel receptionist, I want to cancel an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	8
	As a hotel receptionist, I want to check room availability, so that I can advise a customer whether there is availability for their chosen date(s).	*Unassigned*	Catherine McIlroy	🔺	TO DO	*Unresolved*	6
	As a customer with a newly made reservation, I want to pay for that reservation, so that there is no outstanding bill.	*Unassigned*	Catherine McIlroy	=	TO DO	*Unresolved*	8
	As a hotel receptionist, I want to check guests in or out of the hotel, so that there is a current record of all guests staying in the hotel.	*Unassigned*	Catherine McIlroy	=	TO DO	*Unresolved*	5
	As a hotel receptionist, I want to confirm payment made by a customer, so that the customer will receive a confirmation email.	*Unassigned*	Catherine McIlroy	=	TO DO	*Unresolved*	2
	As a customer with past bookings, I want to view my reservation history, so that I can see the details of my previous reservations.	*Unassigned*	Catherine McIlroy	🔻	TO DO	*Unresolved*	7
	As a hotel manager, I want to add or remove rooms from the room inventory, so that the room inventory is kept up to date.	*Unassigned*	Catherine McIlroy	🔻	TO DO	*Unresolved*	5
	As a hotel manager, I want to generate periodical reports, so that I can identify trends in reservations made.	*Unassigned*	Catherine McIlroy	⏬	TO DO	*Unresolved*	4

This can also be displayed in simple table format, as shown below.

Issue Type	Summary	Assignee	Assignee Id	Reporter	Priority	Status	Resolution	Story points
Story	As a system user, I want to be able to log in to the reservation system, so that I can use the features of the system.			Catherine McIlroy	Highest	To Do		5.0
Story	As a system user, I want to be able to log out of the reservation system, so that my account is private and secure.			Catherine McIlroy	Highest	To Do		5.0
Story	As a customer, I want to make a new reservation, so that I can stay at the hotel.			Catherine McIlroy	Highest	To Do		7.0
Story	As a customer considering making a reservation, I want to check current room availability, so that I can see if there is availability for my chosen date(s).			Catherine McIlroy	Highest	To Do		6.0
Story	As a hotel receptionist, I want to confirm a newly made reservation, so that the customer receives booking confirmation.			Catherine McIlroy	Highest	To Do		2.0
Story	As a customer with an existing reservation, I want to modify my reservation, so that it meets the needs of my changed plans.			Catherine McIlroy	High	To Do		7.0
Story	As a customer with an existing reservation, I want to cancel my reservation, so that it meets the needs of my changed plans.			Catherine McIlroy	High	To Do		5.0
Story	As a hotel receptionist, I want to make a new reservation on behalf of a customer, so that the customer can stay at the hotel.			Catherine McIlroy	High	To Do		7.0
Story	As a hotel receptionist, I want to modify an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.			Catherine McIlroy	High	To Do		8.0
Story	As a hotel receptionist, I want to cancel an existing booking on behalf of a customer, so that it meets the needs of the customer's changed plans.			Catherine McIlroy	High	To Do		8.0
Story	As a hotel receptionist, I want to check room availability, so that I can advise a customer whether there is availability for their chosen date(s).			Catherine McIlroy	High	To Do		6.0
Story	As a customer with a newly made reservation, I want to pay for that reservation, so that there is no outstanding bill.			Catherine McIlroy	Medium	To Do		8.0
Story	As a hotel receptionist, I want to check guests in or out of the hotel, so that there is a current record of all guests staying in the hotel.			Catherine McIlroy	Medium	To Do		5.0
Story	As a hotel receptionist, I want to confirm payment made by a customer, so that the customer will receive a confirmation email.			Catherine McIlroy	Medium	To Do		2.0
Story	As a customer with past bookings, I want to view my reservation history, so that I can see the details of my previous reservations.			Catherine McIlroy	Low	To Do		7.0
Story	As a hotel manager, I want to add or remove rooms from the room inventory, so that the room inventory is kept up to date.			Catherine McIlroy	Low	To Do		5.0
Story	As a hotel manager, I want to generate periodical reports, so that I can identify trends in reservations made.			Catherine McIlroy	Lowest	To Do		4.0

Prior to each sprint, a few user stories are selected from the product backlog and added to the sprint backlog. This is a list of features to be completed in the current sprint. An example of the sprint backlog for this project prior to commencement of the first sprint is included below.

.

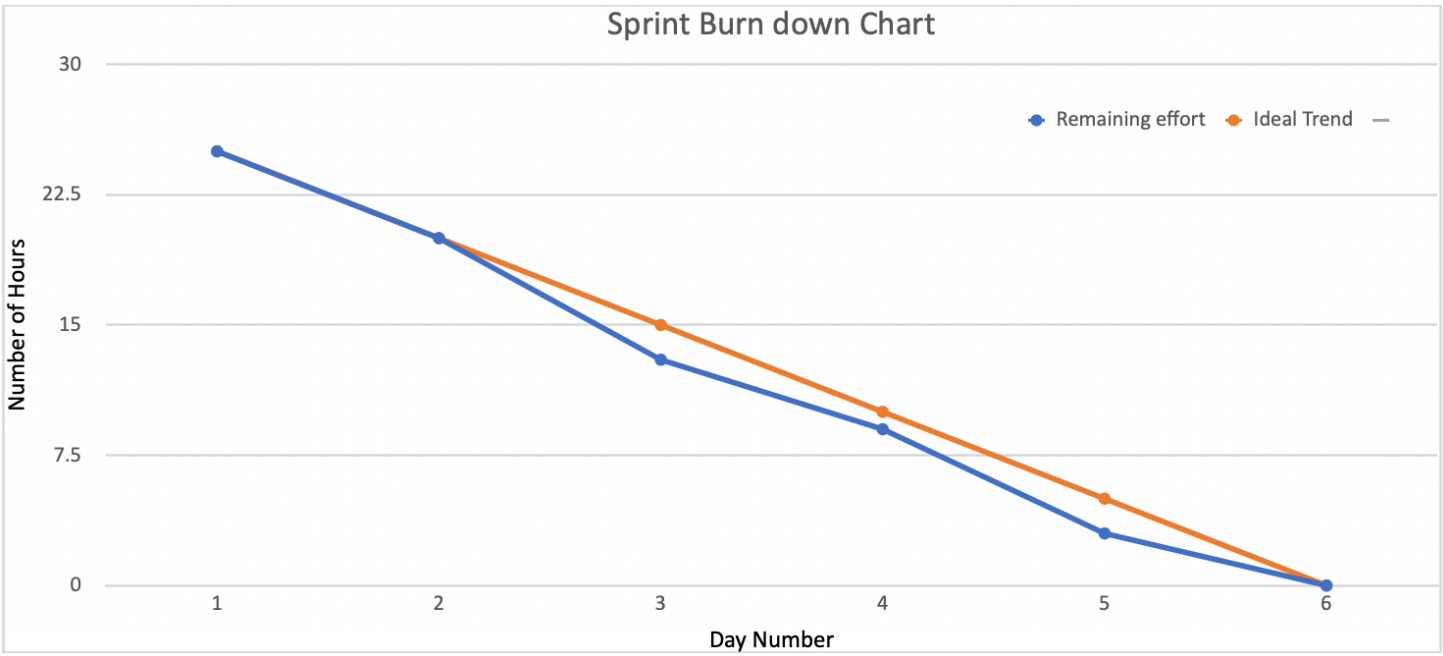
T	Summary	Assignee	Reporter	P	Status	Resolution	Story point estimate
	As a system user, I want to be able to log in to the reservation system, so that I can use the features of the system.	*Unassigned*	Catherine McIlroy		TO DO	*Unresolved*	5
	As a system user, I want to be able to log out of the reservation system, so that my account is private and secure.	*Unassigned*	Catherine McIlroy		TO DO	*Unresolved*	5
	As a customer, I want to make a new reservation, so that I can stay at the hotel.	*Unassigned*	Catherine McIlroy		TO DO	*Unresolved*	7
	As a customer considering making a reservation, I want to check current room availability, so that I can see if there is availability for my chosen date(s).	*Unassigned*	Catherine McIlroy		TO DO	*Unresolved*	6
	As a hotel receptionist, I want to confirm a newly made reservation, so that the customer receives booking confirmation.	*Unassigned*	Catherine McIlroy		TO DO	*Unresolved*	2

As can be seen above, this sprint backlog includes the highest priority tasks from the product backlog, and details user stories which must be completed in order for the hotel reservation system to have basic functionality.

The final artifact which will be included in this chapter is the burndown chart for Sprint 1. The below table displays the names of the features being worked on, the ideal number of hours to be spent on each of these features each day, and the actual number of hours which were spent each day. The associated chart displays the trend over time of actual hours spent vs the estimated (ideal) time.

Sprint 1 Burn down Chart							
		Initial Estimate hours					
Backlog ID	User Stories	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5
1	Log In	5	1	2	1	1	0
2	Log out	5	1	1	1	1	1
3	Make new reservation	7	1	2	1	1	2
4	Check room availability	6	2	2	1	1	0
5	Receptionist confirm reservation	2	0	0	0	2	0
Remaining effort		25	20	13	9	3	0
Ideal Trend		25	20	15	10	5	0

Table showing ideal trend vs actual trend



Burndown chart displaying the figures in the above table

RISK MANAGEMENT

There are many potential risks associated with software development projects, and this project is no exception. The project risks include the following:

- Client dissatisfaction with finished product
- Missing/incomplete software features
- Exceeding set budget
- Exceeding set time frame for development
- Change of design plans
- Failure to complete the development process

These are displayed in the risk matrix below along with their corresponding likelihood and severity.

	Date	Project Name:		Project Manager		
	25/04/2024	Hotel Reservation System		Catherine McIlroy		
		RISK ASSESSMENT MATRIX				
		Likelihood				
		Very Likely	Likely	Possible	Unlikely	Highly Unlikely (Rare)
Severity	Catastrophic					Failure to complete the development process
	Serious				Client dissatisfaction with finished product	Missing/ incomplete software features
	Moderate			Exceeding budget	Exceeding set time frame for development	
	Negligible		Change of design plans			

Notice from the above matrix that the majority of these risks fall towards the lower end of the likelihood scale. This is due to our project's adherence to agile scrum methodology, which significantly reduces the likelihood of occurrence of the above listed risks.

Because the product is developed in relatively short cycles, or sprints, and a working update is produced at the end of each of these sprints, this means that the client can keep track of development progress and the state of the product. This makes it unlikely that the client would be dissatisfied with the finished product, since any issues would be flagged early in the production cycle.

Features are developed in priority order from highest to lowest, and progress is reviewed on a daily basis, with a focus on completing all features assigned for that particular sprint within the designated 2 week time period. Daily scrum meetings also allow problems to be flagged and resolved promptly in order to prevent delays. These aspects of the scrum framework mean that it is highly unlikely that the finished product would be missing any features, or that the development process would not be completed.

Again, due to the repeated relatively short development cycles, it is much easier to manage budgets and deadlines, so the likelihood of exceeding either of these is quite low.

Lastly, while it is quite likely that client requirements for the product could change throughout the development cycle, the use of sprint cycles, release of product increments and continuous product review mean that the severity of this is negligible. The client can make alterations to their product requirements prior to the start of the next sprint, and the team can work these changes into the next iteration of the product.

QUALITY MANAGEMENT

The adherence of the project to an agile approach allows for continuous monitoring of product quality throughout the development cycle. Product testing is a part of every sprint cycle, which allows defects to be identified and fixed at an early stage. It is also much easier to detect and fix issues in a small product update than it would be to do so for a large finished product at the end of the project.

Product development and work progress is tracked by implementing daily scrum meetings and team discussion of results. Each sprint ends with a "Sprint Retrospective", where the team discusses what went well in the sprint, what could be improved, and what improvements might be implemented in the next sprint. These meetings and discussions allow for constant reflection on how product quality can be improved.

Agile also focuses on the client's needs, and the client has constant input throughout the entirety of the project. This makes it much more likely that the finished product will meet the client's standards.

COMMUNICATION MANAGEMENT

The communication matrix below details the strategies used throughout this project to ensure good, efficient communication between all members of the development team, and also between the project team and the stakeholders.

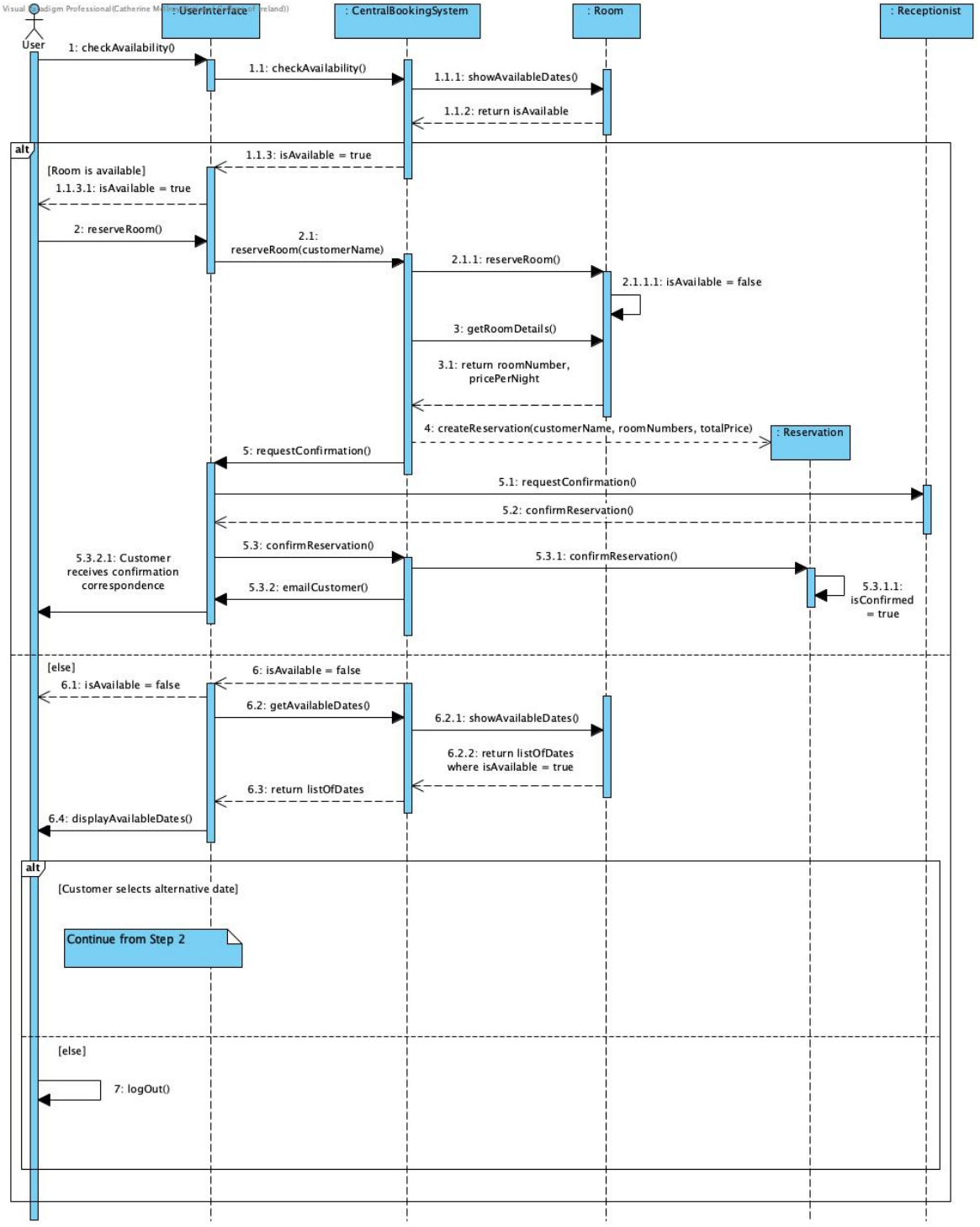
Communication Matrix

COMMUNICATION	PURPOSE	MEDIUM	FREQUENCY	AUDIENCE
Sprint Planning	Set a sprint goal. Take features from product backlog and add to sprint backlog	In person	Start of every sprint	Product Owner Development Team Scrum Master
Daily Scrum	Discuss progress, identify and rectify any problems	In person	Daily	Product Owner Development Team Scrum Master
Sprint Review	Report progress made during the previous sprint	In person	End of every sprint	Product Owner Stakeholders Development Team Scrum Master
Sprint Retrospective	Discuss what went well, what could potentially be improved	In person	End of every sprint	Product Owner Development Team Scrum Master

Classes for Implementation of Make Reservation Use Case (Task 12)

This section will detail the classes required to implement the ‘Make Reservation’ use case of the Hotel Reservation System.

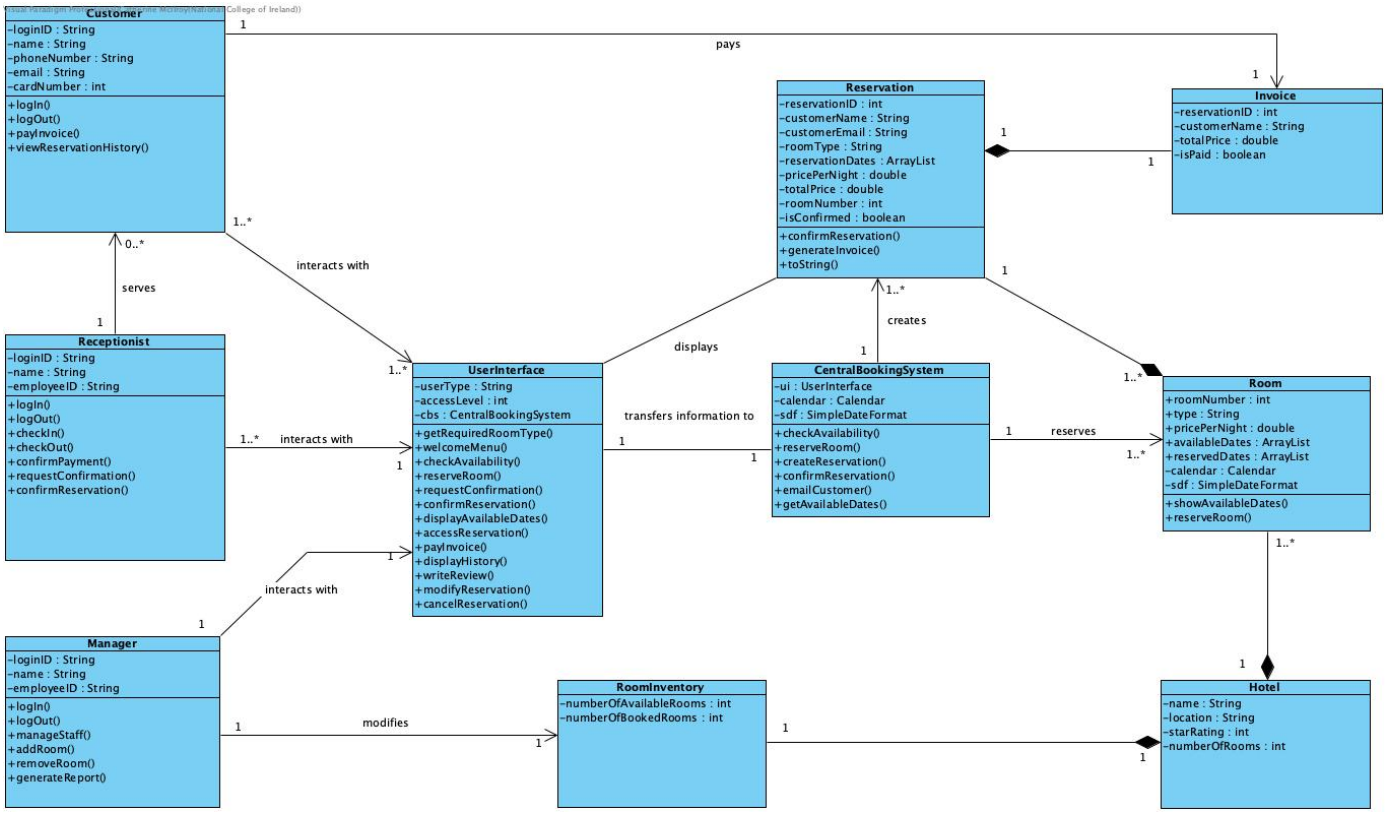
This use case was previously described in CA Part 1. I have included the sequence diagram from Part 1 below as a reminder of the planned operations involved in this use case.



Sequence diagram for the ‘Make Reservation’ use case

Please note that there may be some minor differences between what is displayed in the above diagram, and the actual code.

I have also included the class diagram for the Hotel Reservation System, which is shown below. This has been modified slightly from the diagram which was included in CA Part 1, as the attributes and operations of each class became more apparent once development of the classes began.



Class diagram for the Hotel Reservation System, detailing attributes and operations of each class

The above classes were developed and written in the Java language. The code for each class is included below.

APPLICATION CLASS (MAIN)

```java
public class Application {
    public static void main(String[] args) {
        // instantiate new objects
        UserInterface ui = new UserInterface("Test");
        String roomType = ui.getRequiredRoomType();
        Room room = new Room(roomType);
        ui.welcomeMenu(room);
    }
}
```

USER INTERFACE CLASS

```java
// import necessary classes
import java.util.Scanner;
public class UserInterface{

    // initialise variables
    private CentralBookingSystem cbs;
    private String userType;
    private int accessLevel;

    // constructor
    public UserInterface(String userType){
        this.cbs = new CentralBookingSystem();
        this.userType = userType;
        if(userType.equals("Customer")){
            accessLevel = 1;
        }
        else if(userType.equals("Receptionist")){
            accessLevel = 2;
        }
        else if(userType.equals("Manager")){
            accessLevel = 3;
```

```
        }
        else if(userType.equals("Test")){
            accessLevel = 4;
        }
        else{
            accessLevel = 0;
        }
    }

    public String getRequiredRoomType() {
        Scanner sc = new Scanner(System.in);
        System.out.println("**************************************\n\nWELCOME TO
THE HOTEL RESERVATION SYSTEM\n\n**************************************");
        System.out.println("\n\nBefore we begin, please enter the type of room
you wish to enquire about (Standard Double €130pn / Superior Double €180pn
/ King €220pn / Junior Suite €300pn / Executive Suite €350pn): ");
        String roomType = sc.nextLine();
        if(!(roomType.equalsIgnoreCase("Standard Double")||
roomType.equalsIgnoreCase("Superior Double")||
roomType.equalsIgnoreCase("King")||roomType.equalsIgnoreCase("Junior
Suite")||roomType.equalsIgnoreCase("Executive Suite"))){
            System.out.println("Error: Invalid Input.\nPlease enter a valid room
type.");
            getRequiredRoomType();
        }
        return roomType;
```

```
    }


    public void welcomeMenu(Room room){
        Scanner sc = new Scanner(System.in);
        String userInput;
        do {
            System.out.println("""
                In order for us to accurately direct your request, please select from the following options:\n\n
                1) Reserve Room\n
                2) Check Room Availability\n
                3) Display Available Dates\n\n
                4) Exit
                """);
            userInput = sc.nextLine();
            if (userInput.equals("1") || userInput.equalsIgnoreCase("reserve Room")) {
                reserveRoom(room);
            } else if (userInput.equals("2") || userInput.equalsIgnoreCase("Check Room Availability")) {
                checkAvailability(room);
            } else if (userInput.equals("3") || userInput.equalsIgnoreCase("Display Available Dates")) {
                displayAvailableDates(room);
            } else if (userInput.equals("4") || userInput.equalsIgnoreCase("Exit")) {
```

```
            System.out.println("Thank you for using the Hotel Reservation System.\nWe hope to see you again soon!");
            return;
        } else {
            System.out.println("Error: Invalid Input.\nPlease enter a valid option from the menu.");
        }
    } while (!userInput.equals("4") && !userInput.equalsIgnoreCase("Exit"));
}

////////////////////////////////////////////////////////////////// CHECK ROOM AVAILABILITY //////////////////////////////////////////////////////////////////

// first we need to check if the room is available on the customer's selected dates
// this method will pass the entered dates to the CentralBookingSystem class, retrieve
// and return the boolean value from the CentralBookingSystem class checkAvailability() method
public void checkAvailability(Room room) {
    System.out.println("Hi! 👋\nI'm the Availability Checking Assistant.\nI just need to get a few details from you before I can process your request.\n\n");
    Scanner sc = new Scanner(System.in);
    System.out.println("Please enter check-in date in the format DD-MM-YYYY: ");
```

```
        String startDate = sc.nextLine();

        System.out.println("Please enter check-out date in the format DD-MM-
YYYY: ");

        String endDate = sc.nextLine();

          boolean isAvailable = cbs.checkAvailability(startDate, endDate, room);

        // since this class is a user interface, we need to display a message to the
user

        if(isAvailable){

            System.out.println("The dates(s) you have selected are available.");

        }

        else{

            System.out.println("The date(s) you have selected are unavailable.
\nPlease select 'Display Available Dates' to see a list of all available dates.");

        }

    }



    public boolean checkAvailability(String startDate, String endDate, Room
room){

          boolean isAvailable = cbs.checkAvailability(startDate, endDate, room);

        return isAvailable;

    }



////////////////////////////////////////////////////////////////// RESERVE A
ROOM ////////////////////////////////////////////////////////////////////////
```

```
    // this method will take in the customer's name, the start date and end date of the reservation

    // it will first check that there is availability on these dates by using the checkAvailability method

    // if there is availability, this method will then pass the customer name, start date and end date to

    // the CentralBookingSystem class reserveRoom() method

        public void reserveRoom(Room room) {

        System.out.println("Hi! 👋\nI'm the Reservation Assistant.\nI just need to get a few details from you before I can process your request.\n\n");

        Scanner sc = new Scanner(System.in);

        if(userType.equals("Customer")||userType.equals("Test")){

            System.out.println("Please enter your name: ");

        }

        else if(userType.equals("Receptionist")){

            System.out.println("Please enter the guest name: ");

        }

        String customerName = sc.nextLine();

        System.out.println("Please enter a contact e-mail address: ");

        String customerEmail = sc.nextLine();

        System.out.println("Please enter check-in date in the format DD-MM-YYYY: ");

        String startDate = sc.nextLine();

        System.out.println("Please enter check-out date in the format DD-MM-YYYY: ");
```

```
        String endDate = sc.nextLine();

         // check availability

        boolean isAvailable = checkAvailability(startDate, endDate, room);

        // available, pass details to the CentralBookingSystem

        if(isAvailable){

            cbs.reserveRoom(customerName, customerEmail, startDate, endDate,
room, this.userType);

        }

        else{

            System.out.println("The date(s) you have selected are unavailable.
\nPlease select 'Display Available Dates' to see a list of all available dates.");

        }

    }


////////////////////////////////////////////////////////////////// CONFIRM
RESERVATION ////////////////////////////////////////////////////////////////////


    // this method will accept a Reservation object as a parameter.

    // if user is logged in as a receptionist, request confirmation via user input

    public boolean requestConfirmation(Reservation reservation) {

        boolean isConfirmed = true;

        if(!(userType.equals("Test"))){

        isConfirmed = false;

        }
```

```
        if(userType.equals("Receptionist")) {

            Scanner sc = new Scanner(System.in);

            System.out.println("\n**********\n\nNOTIFICATION\n\n**********\n\nA
new reservation has been made:\n\n" + reservation.toString() + "\n\nConfirm?
\nY/N");

            String userInput = sc.nextLine(); // Store the user input

            // if user input is neither "Y" nor "N", display error message and try
again

            while(!(userInput.equalsIgnoreCase("Y") ||
userInput.equalsIgnoreCase("N"))) {

                System.out.println("Error: Invalid Input.\nPlease enter either 'Y' to
confirm, or 'N' to deny.\nNew reservation: " + reservation.toString() +
"\nConfirm? Y/N");

                userInput = sc.nextLine(); // Prompt again and store the new input

            }

            // if receptionist confirms booking, set isConfirmed = true

            if(userInput.equalsIgnoreCase("Y")) {

                isConfirmed = true;

            }

            // if receptionist rejects booking, set isConfirmed = false

            else {

                isConfirmed = false;

            }

        }

        else if(userType.equals("Customer")){
```

```
            System.out.println("Awaiting action by the hotel.\nYou will be notified
once the reservation has been confirmed.");

        }

        return isConfirmed;

    }


    // this method will accept the result of the confirmation process, and the
reservation details.

    // It will display a message to the user to let them know the outcome of the
confirmation process.

    public void confirmReservation(boolean isConfirmed, Reservation
reservation) {

          if(isConfirmed){

            System.out.println("** USER INTERFACE NOTIFICATION **\n\nYour
booking has been confirmed.\nPlease find details of your reservation
below\n\n" + reservation.toString());

        }

        else{

            System.out.println("** USER INTERFACE NOTIFICATION **\n\nThe
reservation detailed below has been denied.\nPlease contact the hotel at your
earliest convenience.\n\n" + reservation.toString());

        }

    }


////////////////////////////////////////////////////////////////// DISPLAY AVAILABLE
DATES //////////////////////////////////////////////////////////////////
```

```
    // this method will call the showAvailableDates() method of the
CentralBookingSystem class

    // and print a list of available dates to the console

    public void displayAvailableDates(Room room) {

        System.out.println("Current Availability: \n\nAvailable Dates:\n" +
cbs.getAvailableDates(room) + "\n\nUnavailable Dates:\n" +
room.reservedDates);

    }


////////////////////////////////// THE FOLLOWING METHODS ARE NOT REQUIRED
FOR THE MAKE RESERVATION USE
CASE ////////////////////////////////////////////////////////////


    // The following methods have not been developed as they are not required
for this use case.

    // They are included for display purposes only. All are listed as return type
void, this does not

    // reflect the final return type once the methods have been developed


    public void accessReservation(){

        // not required

    }


    public void payInvoice(){

        // not required

    }
```

```
    public void displayHistory(){
        // not required
    }

    public void writeReview(){
        // not required
    }

    public void modifyReservation(){
        // not required
    }

    public void cancelReservation(){
    // not required
    }
}
```

CENTRAL BOOKING SYSTEM CLASS

```java
// import necessary classes
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.text.ParseException;

public class CentralBookingSystem {

    private Calendar calendar = new GregorianCalendar();
    private SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
    private UserInterface ui;

    // constructor
    public CentralBookingSystem(){
    }

////////////////////////////////////////////////////////////////// CHECK ROOM AVAILABILITY //////////////////////////////////////////////////////////////////

    // this method will take in two dates as parameters, startDate and endDate
```

```
// it will get the list of available dates from the Room class and compare
// these to the list of dates for reservation
// it will return a boolean value is Available
public boolean checkAvailability(String start, String end, Room room) {
// initialise isAvailable as true
    boolean isAvailable = true;
    // get list of available dates from Room class
    ArrayList<String> availableDates = room.showAvailableDates(room);
    // create list of dates to be reserved, inclusive of startDate and endDate
    ArrayList<String> reservationDates = new ArrayList<>();
    try {
        Date startDate = sdf.parse(start);
        Date endDate = sdf.parse(end);
        calendar.setTime(startDate);

        while (calendar.getTime().before(endDate)){
            Date date = calendar.getTime();
            String formattedDate = sdf.format(date);
            reservationDates.add(formattedDate);
            calendar.add(Calendar.DATE, 1);
        }
    }
    catch (ParseException e) {
```

```
        }
        // compare dates to be reserved with list of available dates
        // for each date in list of dates to be reserved, if it is not
        // contained in the list of available dates, the room is not available
        // for the entirety of the desired reservation period, set isAvailable = false
        if(!(availableDates.containsAll(reservationDates))){
            isAvailable = false;
        }
        return isAvailable;
    }

////////////////////////////////////////////////////////////////// RESERVE A ROOM ////////////////////////////////////////////////////////////////////

    // this method will accept the customer name, start date and end date of reservation
    // from the UserInterface class. It will create a list of dates inclusive of the start
    // and end dates, and pass this list to the Room class reserveRoom() method.
    // The customer name will be passed to the createReservation method.
    public void reserveRoom(String customerName, String customerEmail, String start, String end, Room room, String userType) {
        // create list of dates to be reserved, inclusive of startDate and endDate
          ArrayList<String> reservationDates = new ArrayList<String>();
```

```
        try {
            Date startDate = sdf.parse(start);
            Date endDate = sdf.parse(end);
            calendar.setTime(startDate);
            while (calendar.getTime().before(endDate)){
                Date date = calendar.getTime();
                String formattedDate = sdf.format(date);
                reservationDates.add(formattedDate);
                calendar.add(Calendar.DATE, 1);
            }
            // pass customer name and reservation dates to the
createReservation() method
            createReservation(customerName, customerEmail, reservationDates,
startDate, endDate, room, userType);
        }
        catch (ParseException e) {
        }
    }


////////////////////////////////////////////////////////////////// CREATE NEW
RESERVATION ////////////////////////////////////////////////////////////////////


    // this method will create a new instance of the Reservation class, with the
attributes
```

```
    // customer name, reservation dates, room number, room type, price per night and total price.


    public Reservation createReservation(String customerName, String customerEmail, ArrayList<String> reservationDates, Date startDate, Date endDate, Room room, String userType) {

        String name = customerName;

        String email = customerEmail;

        ArrayList<String> dates = reservationDates;

        int roomNumber = room.roomNumber;

        String roomType = room.type;

        double pricePerNight = room.pricePerNight;

        // total price is pricePerNight multiplied by number of reservation dates -1 as the guest

        // does not stay overnight on the last date

        double totalPrice = pricePerNight * (reservationDates.size());

        // instantiate new Reservation object with the specified attributes

        Reservation reservation = new Reservation(name, email, dates, roomNumber, roomType, pricePerNight, totalPrice);

        // pass reservation details to confirmReservation method

        confirmReservation(reservation, startDate, endDate, room, userType);

        return reservation;

    }


////////////////////////////////////////////////////////////////// CONFIRM RESERVATION //////////////////////////////////////////////////////////////////
```

```
    // this method will communicate with the UserInterface class to trigger request of confirmation

    // by a receptionist. It accepts a Reservation object as a parameter.

    // It will call the confirmReservation() method of the UserInterface class to display the

    // outcome to the customer, and will call emailCustomer() method to send this message as an email

    public void confirmReservation(Reservation reservation, Date startDate, Date endDate, Room room, String userType) {

        ui = new UserInterface(userType);

        boolean isConfirmed = ui.requestConfirmation(reservation);

        // if reservation is confirmed, call confirmReservation method of Reservation class to

        // change reservation isConfirmed attribute to true.

        if(isConfirmed){

            reservation.confirmReservation(reservation);

            // pass this list to the Room class reserveRoom() method

            room.reserveRoom(room, startDate, endDate);

        }

        // call confirmReservation() method of UserInterface class

        ui.confirmReservation(isConfirmed, reservation);

        // call emailCustomer() method, to communicate result of reservation process to customer

        emailCustomer(isConfirmed, reservation);

    }
```

```
////////////////////////////////////////////////////// EMAIL RESERVATION
OUTCOME TO CUSTOMER ////////////////////////////////////////////////////////////////


   public String emailCustomer(boolean isConfirmed, Reservation reservation)
{

        // get email address associated with reservation

       String customerEmail = reservation.customerEmail;

       String customerName = reservation.customerName;

       String message;

       // for the purposes of this project, the email to the customer will be
displayed as

       // a message to the console

       if(isConfirmed){

           message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail +
"\n\nDear " + customerName + ",\n\nYour booking has been confirmed. Please
find details of your reservation below: \n\n"

           + reservation.toString();

       }

       else{

           message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail +
"\n\nDear " + customerName + ",\n\nThe reservation detailed below has been
denied. Please contact the hotel at your earliest convenience.\n\n"

           + reservation.toString();

       }

       System.out.println(message);
```

```
        return message;
    }

////////////////////////////////////////////////////////////////// DISPLAY AVAILABLE
DATES //////////////////////////////////////////////////////////////////

    // this method will call the showAvailableDates() method of the Room class
    // and will return a list of available dates
    public ArrayList<String> getAvailableDates(Room room) {
         ArrayList<String> availableDates = room.showAvailableDates(room);
        return availableDates;
    }
}
```

ROOM CLASS

```java
// import necessary classes
import java.util.Date;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.text.SimpleDateFormat;
import java.text.ParseException;

public class Room {

    // initialise variables
    public int roomNumber;
    public String type;
    public double pricePerNight;
    public ArrayList<String> availableDates; // Define as instance variable
    public ArrayList<String> reservedDates; // Define as instance variable
    private Calendar calendar = new GregorianCalendar();
    private SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");

    // constructor
    public Room(String type){
```

```
        this.roomNumber = ((int)Math.floor(Math.random() * 100) + 1);
        this.type = type;
        if(type.equalsIgnoreCase("Superior Double")){
            this.pricePerNight = 180;
        }
        else if(type.equalsIgnoreCase("King")){
            this.pricePerNight = 220;
        }
        else if(type.equalsIgnoreCase("Junior Suite")){
            this.pricePerNight = 300;
        }
        else if(type.equalsIgnoreCase("Executive Suite")){
            this.pricePerNight = 350;
        }
        else{
            pricePerNight = 130;
        }

        // initialise availableDates as being all dates between 01/05/2024 and
01/05/2025
        this.availableDates  = new ArrayList<>();
        try {
            Date startDate = sdf.parse("01-05-2024");
            Date endDate = sdf.parse("01-05-2025");{
```

```
        calendar.setTime(startDate);
        while (calendar.getTime().before(endDate)){
            Date date = calendar.getTime();
            String formattedDate = sdf.format(date);
            availableDates.add(formattedDate);
            calendar.add(Calendar.DATE, 1);
        }
    }
}
catch (ParseException e) {


}

this.reservedDates = new ArrayList<>();
try {
    Date startDate = sdf.parse("22-04-2024");
    Date endDate = sdf.parse("01-05-2024");
    calendar.setTime(startDate);
    while (calendar.getTime().before(endDate)){
        Date date = calendar.getTime();
        String formattedDate = sdf.format(date);
        this.reservedDates.add(formattedDate);
        calendar.add(Calendar.DATE, 1);
```

```
            }
        }
        catch (ParseException ex) {
        }
    }

////////////////////////////////////////////////////////////////// DISPLAY ROOM AVAILABILITY //////////////////////////////////////////////////////////////////

    // this method will return an ArrayList of all currently available dates
    public ArrayList<String> showAvailableDates(Room room) {
        return room.availableDates;
    }

////////////////////////////////////////////////////////////////// RESERVE A ROOM //////////////////////////////////////////////////////////////////

    // this method will accept the list of reservation dates from the CentralBookingSystem class reserveRoom() method.
    // It will add these dates to the reservedDates list, and remove them from the availableDates list.
    public void reserveRoom(Room room, Date startDate, Date endDate) {
        calendar.setTime(startDate);
        while (calendar.getTime().before(endDate)){
```

```
            // Create a new instance of Date within the loop
            Date date = calendar.getTime();
            String formattedDate = sdf.format(date);
            room.reservedDates.add(formattedDate);
            room.availableDates.remove(formattedDate);
            calendar.add(Calendar.DATE, 1);
        }
    }
}
```

RESERVATION CLASS

```java
import java.util.ArrayList;
import java.text.DecimalFormat;


public class Reservation {


    // initialise variables
    public int reservationID;
    public String customerName;
    public String customerEmail;
    public String roomType;
    public ArrayList<String> reservationDates;
    public double pricePerNight;
    public double totalPrice;
    public int roomNumber;
    public boolean isConfirmed;


    // constructor
    public Reservation(String customerName, String customerEmail, ArrayList<String> reservationDates, int roomNumber, String roomType, double pricePerNight, double totalPrice){
        this.reservationID = (int)Math.floor((Math.random() * 10000) + 1);
        this.customerName = customerName;
```

```
        this.customerEmail = customerEmail;
        this.reservationDates = reservationDates;
        this.roomNumber = roomNumber;
        this.roomType = roomType;
        this.pricePerNight = pricePerNight;
        this.totalPrice = totalPrice;
        this.isConfirmed = false;
    }

////////////////////////////////////////////////////////////////// CONFIRM
RESERVATION ////////////////////////////////////////////////////////////////////

    // this method will take in a Reservation object as a parameter and change
the isConfirmed attribute to true
    public void confirmReservation(Reservation reservation){
        reservation.isConfirmed = true;
    }

////////////////////////////////////////////////////////////////// OVERRIDE TO
STRING ////////////////////////////////////////////////////////////////////

    // this method will output the reservation details as a String
    @Override
    public String toString(){
```

```
        DecimalFormat f = new DecimalFormat("##.00");
        String str = "";
        str += "Reservation Details\n\n********************\n\n";
        str += "Reservation ID: " + this.reservationID;
        str += "\nName: " + this.customerName;
        str += "\nE-mail: " + this.customerEmail;
        str += "\nDates: " + this.reservationDates;
        str += "\nRoom Number: " + this.roomNumber;
        str += "\nRoom Type: " + this.roomType;
        str += "\nPrice Per Night: €" + f.format(this.pricePerNight);
        str += "\n\nTotal Price: €" + f.format(this.totalPrice);
        return str;
    }

////////////////////////////////// THE FOLLOWING METHODS ARE NOT REQUIRED
FOR THE MAKE RESERVATION USE
CASE ////////////////////////////////////////////////////////////////

    // The following methods have not been developed as they are not required
for this use case.
    // They are included for display purposes only. All are listed as return type
void, this does not
    // reflect the final return type once the methods have been developed

    public void generateInvoice(){
```

```
        // not required
    }
}
```

Test Cases (Task 13)

Testing was planned and carried out for each of the classes described in the previous section.

This section will detail the testing of the CentralBookingSystem class.

As part of the system testing process, unit tests were carried out for each method in the class. The aim of this process was to identify defects in the running of the program. The results of these test cases are detailed below.

PREPARED BY		CATHERINE MCILROY					COMPANY LOGO SoftwareTestingMaterial		
DATE		25/04/2024							
SCENARIO ID	1	SCENARIO DESCRIPTION		CHECK AVAILABILITY OF ROOM ON SPECIFIC DATES					
S.NO	TEST CASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULT	POSTCONDITION	ACTUAL RESULT	STATUS	DEFECT ID
1	TC_CHKAVLBLTY_1	Enter date range which falls fully within unavailable dates.	User is logged into the Hotel Reservation System and selected option ‘2’ from the welcome menu	start = "22-04-2024"; end = "30-04-2024";	isAvailable = false	Availability status of room on entered dates is ascertained	isAvailable = fals	PASS	N/A
1	TC_CHKAVLBLTY_2	Enter date range which falls on both available and unavailable dates.	User is logged into the Hotel Reservation System and selected option ‘2’ from the welcome menu	start = "30-04-2024"; end = “02-05-2024";	isAvailable = false	Availability status of room on entered dates is ascertained	isAvailable = fals	PASS	N/A
1	TC_CHKAVLBLTY_3	Enter date range which falls fully within available dates.	User is logged into the Hotel Reservation System and selected option ‘2’ from the welcome menu	start = "01-05-2024"; end = “06-05-2024”;	isAvailable = true	Availability status of room on entered dates is ascertained	isAvailable = true	PASS	N/A

Test Case 1- Check availability of room on specific dates. Boundary testing.

PREPARED BY		CATHERINE MCILROY							
DATE		25/04/2024						COMPANY LOGO SoftwareTestingMaterial	
SCENARIO ID	2	SCENARIO DESCRIPTION		RESERVE A ROOM					
S.NO	TEST CASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULT	POSTCONDITION	ACTUAL RESULT	STATUS	DEFECT ID
2	TC_RSVRM_1	User type is “Receptionist”, user input “N” when prompted to confirm reservation	User is logged into the Hotel Reservation System and selected option ‘1’ from the welcome menu	room = new Room("Standard Double"); customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; start = "01-05-2024"; end = "06-05-2024"; userType = "Receptionist";	Reservation denied (i.e. false)	Room has not been reserved.	Reservation denied (i.e. false)	PASS	N/A
2	TC_RSVRM_2	User type is “Receptionist”, user input “Y” when prompted to confirm reservation	User is logged into the Hotel Reservation System and selected option ‘1’ from the welcome menu	room = new Room("Standard Double"); customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; start = "01-05-2024"; end = "06-05-2024"; userType = "Receptionist";	Reservation confirmed (i.e. true)	Room has been reserved.	Reservation confirmed (i.e. true)	PASS	N/A
2	TC_RSVRM_3	User type is “Test”, no user input required as reservation will automatically be confirmed	User is logged into the Hotel Reservation System and selected option ‘1’ from the welcome menu	room = new Room("Standard Double"); customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; start = "01-05-2024"; end = "06-05-2024"; userType = “Test”;	Reservation confirmed (i.e. true)	Room has been reserved.	Reservation confirmed (i.e. true)	PASS	N/A

Test Case 2- Reserve a room

PREPARED BY		CATHERINE MCILROY					COMPANY LOGO SoftwareTestingMaterial		
DATE		25/04/2024							
SCENARIO ID	3	SCENARIO DESCRIPTION		CREATE RESERVATION					
S.NO	TEST CASE ID	TEST CASE DESCRIPTION	PRECONDITION	TEST DATA	EXPECTED RESULT	POSTCONDITION	ACTUAL RESULT	STATUS	DEFECT ID
3	TC_CRTRSVN_1	Create reservation for a room type **Standard Double**	A customer has requested to reserve a room (or a receptionist has made this request on behalf of a customer), and the room is available on the requested dates.	userType = "Test" customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Standard Double")	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Standard Double" pricePerNight = 130 totalPrice = 650	A new Reservation object has been created	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Standard Double" pricePerNight = 130 totalPrice = 650	PASS	N/A
3	TC_CRTRSVN_2	Create reservation for a room type **Superior Double**	A customer has requested to reserve a room (or a receptionist has made this request on behalf of a customer), and the room is available on the requested dates.	userType = "Test" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Superior Double")	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Superior Double" pricePerNight = 180 totalPrice = 900	A new Reservation object has been created	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Superior Double" pricePerNight = 180 totalPrice = 900	PASS	N/A

Test Case 3- Create a new reservation. Testing for each room type (continued on next page)

3	TC_CRTRSVN_3	Create reservation for a room type **King**	A customer has requested to reserve a room (or a receptionist has made this request on behalf of a customer), and the room is available on the requested dates.	userType = "Test" customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("King")	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "King" pricePerNight = 220 totalPrice = 1100	A new Reservation object has been created	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "King" pricePerNight = 220 totalPrice = 1100	PASS	N/A
3	TC_CRTRSVN_4	Create reservation for a room type **Junior Suite**	A customer has requested to reserve a room (or a receptionist has made this request on behalf of a customer), and the room is available on the requested dates.	userType = "Test" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Junior Suite")	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Junior Suite" pricePerNight = 300 totalPrice = 1500	A new Reservation object has been created	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Junior Suite" pricePerNight = 300 totalPrice = 1500	PASS	N/A
3	TC_CRTRSVN_5	Create reservation for a room type **Executive Suite**	A customer has requested to reserve a room (or a receptionist has made this request on behalf of a customer), and the room is available on the requested dates.	userType = "Test" customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Executive Suite")	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Executive Suite" pricePerNight = 350 totalPrice = 1750	A new Reservation object has been created	customerName = "Catherine McIlroy" customerEmail = "x23173190@student.ncirl.ie" reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") roomType = "Executive Suite" pricePerNight = 350 totalPrice = 1750	PASS	N/A

Test Case 3- Create a new reservation (continued)

PREPARED BY		CATHERINE MCILROY						COMPANY LOGO SoftwareTestingMaterial
DATE		25/04/2024						
SCENARIO ID	4	SCENARIO DESCRIPTION		CONFIRM RESERVATION				
S.NO	TEST CASE ID	ST CASE DESCRIPTIO	RECONDITIO	TEST DATA	XPECTED RESU	OSTCONDITIO	ACTUAL RESULT	STATUS	EFECT ID
4	TC_CNFMRSVN_1	User type is “Receptionist”, user input “Y” when prompted to confirm reservation	A new Reservation object has been created	userType = "Receptionist" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Standard Double") reservation = new Reservation(customer Name, customerEmail, reservationDates, startDate, endDate, room, userType)	The reservation is confirmed (i.e. true)	The confirmation status of the Reservation object (boolean isConfirmed) has been modified to true.	The reservation is confirmed (i.e. true)	PASS	N/A
4	TC_CNFMRSVN_2	User type is “Receptionist”, user input “N” when prompted to confirm reservation	A new Reservation object has been created	userType = "Receptionist" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Standard Double") reservation = new Reservation(customer Name, customerEmail, reservationDates, startDate, endDate, room, userType)	The reservation is not confirmed (i.e. false)	The confirmation status of the Reservation object (boolean isConfirmed) has not been changed (i.e. it is false).	The reservation is not confirmed (i.e. false)	PASS	N/A

Test Case 4- Confirm a new reservation (continued on next page)

4	TC_CNFMRSVN_3	User type is “Test”, no user input required as boolean isConfirmed will automatically be set to true	A new Reservation object has been created	userType = “Test" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = "06-05-2024" room = new Room("Standard Double") reservation = new Reservation(customer Name, customerEmail, reservationDates, startDate, endDate, room, userType)	The reservation is confirmed (i.e. true)	The confirmation status of the Reservation object (boolean isConfirmed) has been modified to true.	The reservation is confirmed (i.e. true)	PASS	N/A

Test Case 4- Confirm a new reservation (continued)

PREPARED BY		CATHERINE MCILROY							
DATE		25/04/2024							
SCENARIO ID	5	SCENARIO DESCRIPTION		EMAIL CUSTOMER					
S.NO	TEST CASE ID	TEST CASE DESCRIPTION	RECONDITIO	TEST DATA	EXPECTED RESULT	OSTCONDITIO	ACTUAL RESULT	STATUS	DEFECT
5	TC_EMLCUST_1	A message is sent to the email address associated with the Reservation object to confirm that the reservation has been made.	A reservation has been confirmed (the boolean isConfirmed attribute of a Reservation object has been set to true)	userType = "Test" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = “06-05-2024" room = new Room("Standard Double") reservation = new Reservation(customer Name, customerEmail, reservationDates, startDate, endDate, room, userType)	Message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail + "\n\nDear " + customerName + ", \n\nYour booking has been confirmed. Please find details of your reservation below: \n\n" + reservation.toString()	A message has been sent to the email address associated with the Reservation object to confirm that the reservation has been made.	Message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail + "\n\nDear " + customerName + ",\n\nYour booking has been confirmed. Please find details of your reservation below: \n\n" + reservation.toString()	PASS	N/A
5	TC_EMLCUST_2	A message is sent to the email address associated with the Reservation object to inform the guest that the reservation has been denied.	A reservation has been denied (the boolean isConfirmed attribute of a Reservation object has been set to false)	userType = "Test" customerName = "Catherine McIlroy"; customerEmail = "x23173190@student.ncirl.ie"; reservationDates = ("01-05-2024", "02-05-2024", "03-05-2024", "04-05-2024", "05-05-2024") startDate = "01-05-2024") endDate = “06-05-2024" room = new Room("Standard Double") reservation = new Reservation(customer Name, customerEmail, reservationDates, startDate, endDate, room, userType)	Message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail + "\n\nDear " + customerName + ", \n\nThe reservation detailed below has been denied. Please contact the hotel at your earliest convenience.\n\n" + reservation.toString()	A message has been sent to the email address associated with the Reservation object to inform the guest that the reservation has been denied.	Message = "** E-MAIL NOTIFICATION **\n\n" + customerEmail + "\n\nDear " + customerName + ",\n\nThe reservation detailed below has been denied. Please contact the hotel at your earliest convenience. \n\n" + reservation.toString()	PASS	N/A

Test Case 5- Email customer with result of reservation request

REPARED BY		CATHERINE MCILROY					COMPANY LOGO SoftwareTestingMaterial		
ATE		25/04/2024							
CENARIO ID	6	SCENARIO DESCRIPTION		GET AVAILABLE DATES					
S.NO	TEST CASE ID	TEST CASE DESCRIPTION	RECONDITIO	TEST DATA	EXPECTED RESULT	OSTCONDITIO	ACTUAL RESULT	STATUS	DEFECT ID
6	TC_GETDATES_1	A list of available dates is requested for a specific room.	A Room object has been created.	startDate = "01-05-2024") endDate = “01-05-2025"	List of available dates will be 01-05-2024 to 01-05-2025 inclusive (i.e. ArrayList returned from getAvailableDates() method will match the input ArrayList)	A list of the available dates for a specific room are displayed.	List of available dates is 01-05-2024 to 01-05-2025 inclusive (i.e. ArrayList returned from getAvailableDates() method matches the input ArrayList)	PASS	N/A

Test Case 6- Get available dates for a specific room

As can be seen from the results of these test cases, all tests were passed and no defects were detected in the CentralBookingSystem class.

This does not necessarily mean that there are no defects present, since testing can only detect the presence of defects, not their absence.

Further testing will need to be carried out to ensure that the program will behave in the way in which it is expected to.

The Java code for the test class CentralBookingSystemTest is included below.

TEST CLASS - CENTRAL BOOKING SYSTEM UNIT TESTS

```java
import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import org.junit.Test;
import static org.junit.jupiter.api.Assertions.*;
import java.io.ByteArrayInputStream;
import java.util.Calendar;
import java.util.GregorianCalendar;

/**
 *
 * @author catherinemcilroy
 */
public class CentralBookingSystemTest {

    private CentralBookingSystem cbs = new CentralBookingSystem();
    private SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");

    public CentralBookingSystemTest() {
```

```
    }


    /**
     * Test of checkAvailability method, of class CentralBookingSystem.
     * All dates between 22-04-2024 and 30-04-2024 inclusive should be unavailable.
     * All dates between 01-05-2024 and 01-05-2025 inclusive should be available.
     * There are three different possible scenarios.
     * Test Case 1: date range falls fully within unavailable dates.
     *      Start Date: 22-04-2024
     *      End Date: 30-04-2024
     *      Expected Result: false
     * Test Case 2: date range falls on both available and unavailable dates.
     *      Start Date: 30-04-2024
     *      End Date: 02-05-2024
     *      Expected Result: false
     * Test Case 3: date range falls fully within available dates.
     *      Start Date: 01-05-2024
     *      End Date: 06-05-2024
     *      Expected Result: true
     */
    @Test
    public void testCheckAvailability() {
```

```
        System.out.println("Testing the checkAvailability() method of the
CentralBookingSystem class.");

        Room room = new Room("Standard Double");

        ////////////////////////////////////// TEST CASE
1 //////////////////////////////////////

        String start = "22-04-2024";

        String end = "30-04-2024";

        boolean expResult = false;

        boolean result = cbs.checkAvailability(start, end, room);

        assertTrue(expResult == result);

        ////////////////////////////////////// TEST CASE 2 //////////////////////////////////

        start = "30-04-2024";

        end = "02-05-2024";

        expResult = false;

        result = cbs.checkAvailability(start, end, room);

        assertTrue(expResult == result);

        ////////////////////////////////////// TEST CASE 3 //////////////////////////////////

        start = "01-05-2024";

        end = "06-05-2024";

        expResult = true;

        result = cbs.checkAvailability(start, end, room);

        assertTrue(expResult == result);

    }
```

```
//    /**
//     * Test of reserveRoom() method, of class CentralBookingSystem.
//     * Test Case 1: Run using userType "Receptionist", with user input "N" to deny reservation.
//     * After method has been called, the dates between start and end inclusive should still be available, as
//     * reservation has been denied.
//          * userType: "Receptionist"
//          * User Input: "N"
//          * Expected result: true
//     * Test Case 2: Run using userType "Receptionist", with user input "Y" to confirm reservation.
//     * After method has been called, the dates between start and end inclusive should still be unavailable, as
//     * reservation has been confirmed.
//          * userType: "Receptionist"
//          * User Input: "Y"
//          * Expected result: false
//     * Test Case 3: Run using userType "Test", this will automatically confirm reservation without requiring user input
//     * After the method has been run, the dates between start and end inclusive should be unavailable.
//          * userType: "Test"
//          * Expected result: false
//     */
```

```
    @Test
    public void testReserveRoom() {
        System.out.println("Testing the reserveRoom() method of the
CentralBookingSystem class.");
        Room room = new Room("Standard Double");
        String customerName = "Catherine McIlroy";
        String customerEmail = "x23173190@student.ncirl.ie";
        String start = "01-05-2024";
        String end = "06-05-2024";
        ////////////////////////////////////// TEST CASE
1 //////////////////////////////////////
        mockUserInput("N");
        String userType = "Receptionist";
        cbs.reserveRoom(customerName, customerEmail, start, end, room,
userType);
        assertTrue(cbs.checkAvailability("01-05-2024", "06-05-2024", room));
        ////////////////////////////////////// TEST CASE
2 //////////////////////////////////////
        mockUserInput("Y");
        cbs.reserveRoom(customerName, customerEmail, start, end, room,
userType);
        assertFalse(cbs.checkAvailability("01-05-2024", "06-05-2024", room));
//      ////////////////////////////////////// TEST CASE
3 //////////////////////////////////////
        Room room2 = new Room("Standard Double");
        userType = "Test";
```

```
        cbs.reserveRoom(customerName, customerEmail, start, end, room2,
userType);

        assertFalse(cbs.checkAvailability("01-05-2024", "06-05-2024", room2));

    }

//

//    /**

//     * Test of createReservation method, of class CentralBookingSystem.

//      * Ensure created Reservation matches expected Reservation.

//      * Each new Reservation object is assigned a random reservationID
between 1 and 10,000

//      * Therefore the expected and actual Reservation objects cannot be
compared directly.

//      * We will instead compare various details of the expect and actual
results including:

//          * Customer Name

//          * Customer Email

//          * Reservation Dates

//          * Room Type

//          * Price per Night

//          * Total Price

//     */

    @Test

    public void testCreateReservation() {

        System.out.println("Testing the createReservation() method of the
CentralBookingSystem class.");
```

```
        String customerName = "Catherine McIlroy";

        String customerEmail = "x23173190@student.ncirl.ie";

        ArrayList<String> reservationDates = new
ArrayList<>(Arrays.asList("01-05-2024", "02-05-2024", "03-05-2024",
"04-05-2024", "05-05-2024"));

        try {

        Date startDate = sdf.parse("01-05-2024");

        Date endDate = sdf.parse("06-05-2024");

        Room room = new Room("Standard Double");

        String userType = "Test";

        Reservation reservation = cbs.createReservation(customerName,
customerEmail, reservationDates, startDate, endDate, room, userType);

        assertTrue(customerName.equals(reservation.customerName));

        assertTrue(customerEmail.equals(reservation.customerEmail));

        assertTrue(reservationDates == reservation.reservationDates);

        assertTrue(room.type.equals(reservation.roomType));

        // testing price per night and total price for each room type

        // Standard Double (€130pn) for 5 nights

        // Expected Total Price = €650

        assertTrue(reservation.pricePerNight == 130.00);

        assertTrue(reservation.totalPrice == 650.00);

        // Superior Double (€180pn) for 5 nights

        // Expected Total Price = €900

        Room supRoom = new Room("Superior Double");
```

```
        reservation = cbs.createReservation(customerName, customerEmail,
reservationDates, startDate, endDate, supRoom, userType);

        assertTrue(reservation.pricePerNight == 180.00);

        assertTrue(reservation.totalPrice == 900.00);

        // King (€220pn) for 5 nights

        // Expected Total Price = €1,100

        Room kingRoom = new Room("King");

        reservation = cbs.createReservation(customerName, customerEmail,
reservationDates, startDate, endDate, kingRoom, userType);

        assertTrue(reservation.pricePerNight == 220.00);

        assertTrue(reservation.totalPrice == 1100.00);

        // Junior Suite (€300pn) for 5 nights

        // Expected Total Price = €1,500

        Room junSuite = new Room("Junior Suite");

        reservation = cbs.createReservation(customerName, customerEmail,
reservationDates, startDate, endDate, junSuite, userType);

        assertTrue(reservation.pricePerNight == 300.00);

        assertTrue(reservation.totalPrice == 1500.00);

        // Executive Suite (€350pn) for 5 nights

        // Expected Total Price = €1,750

        Room execSuite = new Room("Executive Suite");

        reservation = cbs.createReservation(customerName, customerEmail,
reservationDates, startDate, endDate, execSuite, userType);

        assertTrue(reservation.pricePerNight == 350.00);

        assertTrue(reservation.totalPrice == 1750.00);
```

```
        } catch (ParseException e) {
        }

    }

//    /**
//     * Test of confirmReservation() method of class CentralBookingSystem.
//     * Test Case 1:
//          * Call method using userType "Receptionist" and user input "Y"
//          * Expected result: true, as user input "Y" has confirmed the booking
//     * Test Case 2:
//          * Call method using userType "Receptionist" and user input "N"
//          * Expected result: false, as user input "N" has denied the booking
//     * Test Case 3:
//          * Call method using userType "Test"
//          * Expected result: true, as no input required and booking automatically confirmed

    @Test
    public void testConfirmReservation() {
        System.out.println("Testing the confirmReservation() method of the CentralBookingSystem class.");
        String customerName = "Catherine McIlroy";
        String customerEmail = "x23173190@student.ncirl.ie";
```

```
        ArrayList<String> reservationDates = new
ArrayList<>(Arrays.asList("01-05-2024", "02-05-2024", "03-05-2024",
"04-05-2024", "05-05-2024"));

        try {

        Date startDate = sdf.parse("01-05-2024");

        Date endDate = sdf.parse("06-05-2024");

        Room room = new Room("Standard Double");

//      ////////////////////////////////////// TEST CASE
1 ////////////////////////////////////////

        String userType = "Receptionist";

        mockUserInput("Y");

        Reservation reservation = cbs.createReservation(customerName,
customerEmail, reservationDates, startDate, endDate, room, userType);

        // make sure isConfirmed is set to false before running
confirmReservation method

        mockUserInput("Y");

        reservation.isConfirmed = false;

        cbs.confirmReservation(reservation, startDate, endDate, room, userType);

        assertTrue(reservation.isConfirmed);

//      ////////////////////////////////////// TEST CASE
2 ////////////////////////////////////////

        mockUserInput("N");

        reservation.isConfirmed = false;

        cbs.confirmReservation(reservation, startDate, endDate, room, userType);

        assertFalse(reservation.isConfirmed);
```

```
//      ////////////////////////////////////// TEST CASE
3 ////////////////////////////////////////
        userType = "Test";
        reservation.isConfirmed = false;
        cbs.confirmReservation(reservation, startDate, endDate, room, userType);
        assertTrue(reservation.isConfirmed);
        }
        catch (ParseException e) {
        }
    }

//    /**
//     * Test of emailCustomer method, of class CentralBookingSystem.
//     * Test Case 1:
//          isConfirmed = true
//          Expected result: confirmedMessage
//     * Test Case 2:
//          isConfirmed = false
//          Expected result: deniedMessage
//     */
    @Test
    public void testEmailCustomer() {
        System.out.println("Testing the emailCustomer() method of the
CentralBookingSystem class.");
```

```
        String customerName = "Catherine McIlroy";

        String customerEmail = "x23173190@student.ncirl.ie";

        ArrayList<String> reservationDates = new
ArrayList<>(Arrays.asList("01-05-2024", "02-05-2024", "03-05-2024",
"04-05-2024", "05-05-2024"));

        try {

        Date startDate = sdf.parse("01-05-2024");

        Date endDate = sdf.parse("06-05-2024");

        Room room = new Room("Standard Double");

        String userType = "Test";

        Reservation reservation = cbs.createReservation(customerName,
customerEmail, reservationDates, startDate, endDate, room, userType);

//      ////////////////////////////////////////////// TEST CASE
1 //////////////////////////////////////////////////

        boolean isConfirmed = true;

        String confirmedMessage = "** E-MAIL NOTIFICATION **\n\n" +
customerEmail + "\n\nDear " + customerName + ",\n\nYour booking has been
confirmed. Please find details of your reservation below: \n\n"

            + reservation.toString();

        String result = cbs.emailCustomer(isConfirmed, reservation);

        assertTrue(confirmedMessage.equals(result));

//      ////////////////////////////////////////////// TEST CASE
2 //////////////////////////////////////////////////

        isConfirmed = false;

        String deniedMessage = "** E-MAIL NOTIFICATION **\n\n" +
customerEmail + "\n\nDear " + customerName + ",\n\nThe reservation detailed
```

```
below has been denied. Please contact the hotel at your earliest convenience.
\n\n"
            + reservation.toString();
        result = cbs.emailCustomer(isConfirmed, reservation);
        assertTrue(deniedMessage.equals(result));
        }
        catch (ParseException e) {
        }
    }


//    /**
//     * Test of getAvailableDates method, of class CentralBookingSystem.
//     * Should return list of dates between 01-05-2024 to 01-05-2025 inclusive
//     * Expected result: true
//     */
    @Test
    public void testGetAvailableDates() {
        System.out.println("Testing the getAvailableDates() method of the
CentralBookingSystem class.");
        Calendar calendar = new GregorianCalendar();
        Room room = new Room("Standard Double");
        ArrayList<String> expResult = new ArrayList<>();
        try {
            Date startDate = sdf.parse("01-05-2024");
```

```
            Date endDate = sdf.parse("01-05-2025");{
                calendar.setTime(startDate);
                while (calendar.getTime().before(endDate)){
                    Date date = calendar.getTime();
                    String formattedDate = sdf.format(date);
                    expResult.add(formattedDate);
                    calendar.add(Calendar.DATE, 1);
                }
            }
        }
        catch (ParseException e) {
        }
        ArrayList<String> result = cbs.getAvailableDates(room);
        assertIterableEquals(expResult, result);


    }


    private void mockUserInput(String input) {
        System.setIn(new ByteArrayInputStream(input.getBytes()));
    }
}
```

References

1. Stack Overflow. (n.d.). *how to get a list of dates between two dates in java*. [online] Available at: https://stackoverflow.com/questions/2689379/how-to-get-a-list-of-dates-between-two-dates-in-java [Accessed 22 Apr. 2024].

2. Atlassian (2019). *Jira Cloud*. [online] Atlassian. Available at: https://www.atlassian.com/software/jira. [Accessed 22 Apr 2024.]

3. Agile Project Management. (2015). *User Stories | Agile*. [online] Available at: https://agile.yakubovsky.com/2015/11/user-stories/ [Accessed 22 Apr. 2024].

4. Schwaber, K. and Sutherland, J. (2020). *Scrum Guide*. [online] Scrumguides.org. Available at: https://scrumguides.org/scrum-guide.html.