Implementation Details

High Level Details

- The application is a Windows Console application.
- It was developed with Microsoft Visual Studio 2015.
- It references NuGet package Json.Net Version 11.02 by NewtonSoft. This package is used for working with JSON. You'll need to install this as it is is used in multiple projects.
- Using a DOS prompt invoke the application with arguments as documented below. You can do this from either the bin/Debug or bin/Release folder. I've included a JSON folder with the content needed to perform a run. Change the JSON content as needed to test out different combinations of Condition/Topography/Date/Room.

Invocation

- The application executable is called: HospitalSimulator.exe.
- It takes one argument.
- The argument can be a single API name or and API with arguments separated by the Pipe "|" character.
- If implemented
- Limited validation is performed on calls to the API.
- Here are sample invocations:

Arguments (API)	**Details**
"Register|Patient1|Flu"	Register Patient1 with Condition = Flu Return the collection of Scheduled Consultations. I chose to return the whole collection so as to see both the new and the existing consults. Note: Wrapping the arguments in quotes is necessary as the pipe character would otherwise redirect output.
"Register|Patient2|Cancer|Head&Neck"	Register Patient2 with Condition = Cancer, Topography = Head & Neck Note: no spaces allowed in Argument. Return the collection of Scheduled Consultations
"Register|Patient3|Cancer|Breast"	Register Patient3 with Condition = Cancer, Topography = Breast Return the collection of Scheduled Consultations
RegisteredPatients	Return the collection of Registered Patients
ScheduledConsultations	Return the collection of Scheduled Consultations

Note: You can redirect the output to a file for later viewing.

- `HospitalSimulator.exe RegisteredPatients > RegPats.json`
- `HospitalSimulator.exe "Register|Patient1|Flu" > Register.json`

Resource Location

JSON Resource files are used to prime the resources used by the application. These resource files must exist in a subfolder from where the EXE resides. The subfolder is called "JSON". The files that are expected there are:

- Consultations.json
- Doctors.json
- Patients.json
- TreatmentMachines.json
- TreatmentRooms.json

Exception Handling

This is an important and often overlooked aspect of software development. It is important to not only communicate properly when things go well. An application must also communicate properly when there are issues.

There is very little exception handling in this application. Given the time constraints I decided to provide a few details here to describe my general approach.

With exception handling you have expected and unexpected exceptions. Luckily, one can plan for both.

- For each entry point to the application there should be exception handling that can act to catch any and all exceptions. This is helpful to implement early to aide in development/debugging.
- Based on some operations you can often identify the types of exceptions that can occur. You should check specifically for these types of exceptions and respond accordingly. These could be recoverable issues where you instruct the user to retry the operation later or non-recoverable where you need to communicate an error condition.
- There are also the unplanned for exceptions. Many of these are caught during development/unit test. Being able to catch exceptions at the earliest entry point provides a good starting point for exception handling. Adding them throughout the processing enables a more granular means for identifying and resolving issues. Propagation of the exceptions back up through the calling hierarchy takes some planning.

Instructions Analysis

Requirements take different forms. The same requirements can be stated in multiple ways. I've found that the manner in which requirements are documented are important to understanding and maintaining these documents. Since requirements change over time the requirements document does also. Writing a maintainable requirements document is as important as writing maintainable code.

Analysis of Resources

This is just a simple restatement of the Resource requirements in a tabular form.

Object	Properties	Comments
Treatment room	unique name Treatment machine	can be equipped with a treatment machine
Treatment machine	unique name capability	capability – only a single capability is allowed • Advanced • Simple
Doctor	name set of roles	set of roles – a Doctor must have at least one role, but may have two. The roles are: • Oncologist • General Practitioner
Patient	name condition	condition – patient must have only one
Condition	diagnosis • Cancer • Flu topography • When diagnosis = Cancer the topography is one of: ○ Breast ○ Head & Neck	
Consultation	patient doctor treatment room consultation date	

Patient Condition-Topography / Resource Cross Reference

This table shows for each Patient/Condition-Topography the Doctors and Treatment Rooms that can be assigned if the resources are available. Expressing requirements like these in a tabular format is often helpful in grasping the requirements and in creating/generating test data.

	Doctors			Rooms				
Condition	John (O)	Anna (GP)	Laura (O, GP)	Room One	Room Two	RoomThree / Treatment MachineA (Advanced)	RoomFour / Treatment MachineB (Advanced)	RoomFive / Treatment MachineC (Simple)
Flu		X	X	X	X	X	X	X
Cancer / Breast	X		X			X	X	X
Cancer / Head & Neck	X		X			X	X	

Test Data Creation

Testing this application thoroughly requires developing test data to prime the resources and drive the processing to hit the various code paths that implement the Use Cases and their Extensions. When developing the test data one must consider things such as:

- Can the test data be shared for different use cases and extensions?
- Can it be used for both Black Box testing and Unit Testing?
- Can the test data be used without change in the future or are there some dependencies? For example, Registering a patient can have different results depending on the test data and when the testing is performed. In this case the current date is important, and unless you want to reset your computer's date you need to account for this in creation of test data. This can be handled in multiple ways. I've done this frequently using SQL and Excel with VB Macros to help generate test data. I've even assisted our Testers in development of their test data to save them time and provide a more robust and well documented set of test data.

Additional Resource Population

The requirements provide initialization of Doctors, TreatmentMachines and TreatmentRooms. Additional initialization will be done for Patients and Consultations.

Patients

Resource Type	Resources

Patients	{
	"Patients": [
	{
	"Name": "Patient_1",
	"RegistrationDate": "20180401",
	"Condition": {
	"Diagnosis": "Flu"
	}
	},
	{
	"Name": "Patient_2",
	"RegistrationDate": "20180402",
	"Condition": {
	"Diagnosis": "Cancer",
	"Topography": "Breast"
	}
	},
	{
	"Name": "Patient_3",
	"RegistrationDate": "20180403",
	"Condition": {
	"Diagnosis": "Cancer",
	"Topography": "Head & Neck"
	}
	}
	]
	}

Note: This is just a sample. Various combinations of patients/consultations must be in place to test the different use cases and extensions.

Consultations

Resource Type	Resources

Consultations	{
	"Consultations": [
	{
	"Patient": "Patient_1",
	"PatientCondition": "Cancer",
	"PatientTopography": "Head & Neck",
	"Doctor": "John",
	"DoctorRole": "Oncologist",
	"TreatmentRoom": "RoomThree",
	"MachineName": "MachineA",
	"MachineCapability": "Advanced",
	"ConsultationDate": "20180425"
	},
	{
	"Patient": "Patient_2",
	"PatientCondition": "Cancer",
	"PatientTopography": "Head & Neck",
	"Doctor": "John",
	"DoctorRole": "Oncologist",
	"TreatmentRoom": "RoomThree",
	"MachineName": "MachineA",
	"MachineCapability": "Advanced",
	"ConsultationDate": "20180426"
	},
	{
	"Patient": "Patient_3",
	"PatientCondition": "Cancer",
	"PatientTopography": "Head & Neck",
	"Doctor": "Laura",
	"DoctorRole": "Oncologist",
	"TreatmentRoom": "RoomFour",
	"MachineName": "MachineB",
	"MachineCapability": "Advanced",
	"ConsultationDate": "20180425"
	}
	]
	}

Note: This is just a sample. Various combinations of patients/consultations must be in place to test the different use cases and extensions.

Use Cases

The following use cases cover the requirements of this exercise.

Use Case	Title	Description (High Level)

1	Patient Registration – Condition = Flu	This use case will test for:
		• Registration of patient
		• Consultation added with the following criteria:
		  ○ Doctor must have Role = General Practitioner. Based on given resources, this could be either Anna or Laura.
		  ○ Must occur in a Treatment Room. Since there is no requirement for a specific type of Treatment Room (with or without a Treatment Machine), all Treatment Rooms are available.
		  ○ First available date where both a General Practitioner and Treatment Room are open.
		• New Consultation will be returned
		
		Note: For each of the Patient Registration use cases there would be multiple extensions to cover the various combinations of open slots based on:
		• Patient Condition / Patient Topography: Room and Doctor Availability
		• Date: Room and Doctor availability

2	Patient Registration – Condition = Cancer (Head & Neck)	This use case will test for:
		• Registration of patient
		• Consultation added with the following criteria:
		  ○ Doctor must have Role = Oncologist. Based on given resources, this could be either John or Laura.
		  ○ Must occur in a Treatment Room with Capability = Advanced. This includes RoomThree and RoomFour.
		  ○ First available date where both an Oncologist and Treatment Room with Capability = Advanced are open.
		• New Consultation will be returned

3	Patient Registration – Condition = Cancer (Breast)	This use case will test for:
		• Registration of patient
		• Consultation added with the following criteria:
		  ○ Doctor must have Role = Oncologist. Based on given resources, this could be either John or Laura.
		  ○ Must occur in a Treatment Room with Capability = Advanced or Simple. This includes RoomThree, RoomFour and RoomFive.
		  ○ First available date where both an Oncologist and Treatment Room with Capability = Advanced or Simple are open.
		• New Consultation will be returned

4	Get the List of Registered Patients	This use case will test for: • Retrieving the list of Registered Patients. • Each Patient will be returned as a JSON object. Note: There is no persistence in this application. Only those patients initially loaded or registered during the run of the application will be retrieved.
5	Get the List of Scheduled Consultations	This use case will test for: • Retrieving the set of Scheduled Consultations. • Each Scheduled Consultation will be returned as a JSON object. Note: There is no persistence in this application. Only those consultations initially loaded or created via patient registration during the run of the application will be retrieved.

Other Topics

There are other topics I could have touched on with respect to documentation, code, design, naming conventions and documentation. We can discuss that after you've had a chance to review my work.