Team Doctor

Java Enterprise Architect Certified Master Assignment - Instruction File

Team Doctor

1 Background

The health care industry has seen the widespread adoption of information technology as an enabling technology in recent years. One area of health care with massive potential is using IT to get more doctors / consultants to review a patient's case history. Team Doctor is a not-for-profit company owned by twenty (20) major hospitals located throughout North America, Europe, and Asia. These hospitals have decided to make their specialist consultant resources available to each other at a nominal cost, in order for all hospitals to provide better care to their patients. Consultants will be able to remotely review X-rays, MRI scans, a complete patient history including previous prescriptions and provide feedback to the doctor in charge of the patient on potential avenues of treatment, alternative treatments that should be considered and so on. The 20 hospitals involved can commit to building Team Doctor as all of their patient care and diagnosis systems have been brought online through significant investment in IT over the last 20 years. The hospitals have calculated that the Team Doctor system will improve access to top-level consultants for an average of 78% of cases that require it, while reducing the cost of access to these specialized resources by 67%. These cost savings can then be reinvested in improving other parts of the hospital, especially primary care.

2 Workshop Output

You are the architect for the Team Doctor project. You have been tasked by the hospital's management committee to lead the team responsible for the design, implementation, and ongoing management of the complete system as a turnkey or complete solution.

After an intensive series of discovery workshops with in-house business analysts and subject matter experts, you know the following facts:

- Doctors will place requests for assistance into the Team Doctor system, attaching the patient ID and all relevant supporting documentation / clinical notes.
- The Team Doctor system will place the request for assistance into the work queue for a set of qualified consultants / peers to review and provide assistance to the requesting doctor.
- There are three key main systems to provide access to — (1) the rich content repository, which contains high-resolution medical images (2) the patient record system, which holds the complete medical history of patients and (3) the prescriptions database, which stores all prescriptions issued to patients.
- All three systems provide a web services interface to read data; none of the three systems will be written to — teams of doctors and consultants working remotely will consume data in read-only mode, reach a consensus and store notes and recommendations in the Team Doctor system.
- Any recommendations that are implemented will be recorded by the system controlling that part of the hospital — for example, X-rays and MRI scans will be loaded into the rich content repository, prescriptions will be recorded in the prescriptions database and so on.
- All of these systems are accessed by a single, unique patient ID, guaranteed to be unique across all participating hospitals.
- The hospitals involved are major trauma centers and also operate in multiple time zones and, therefore, the system must be designed with resilience in mind and an availability of 99.99%.
- Patient security is important — doctors will be able to access the system using HTTP over SSL via the Internet, but you should consider what additional measures are warranted to protect patient data.

2.1 Business Domain Model

The following business domain model describes the key objects identified during the workshops. All of these objects and the relationships between them should be addressed in your design and implementation.

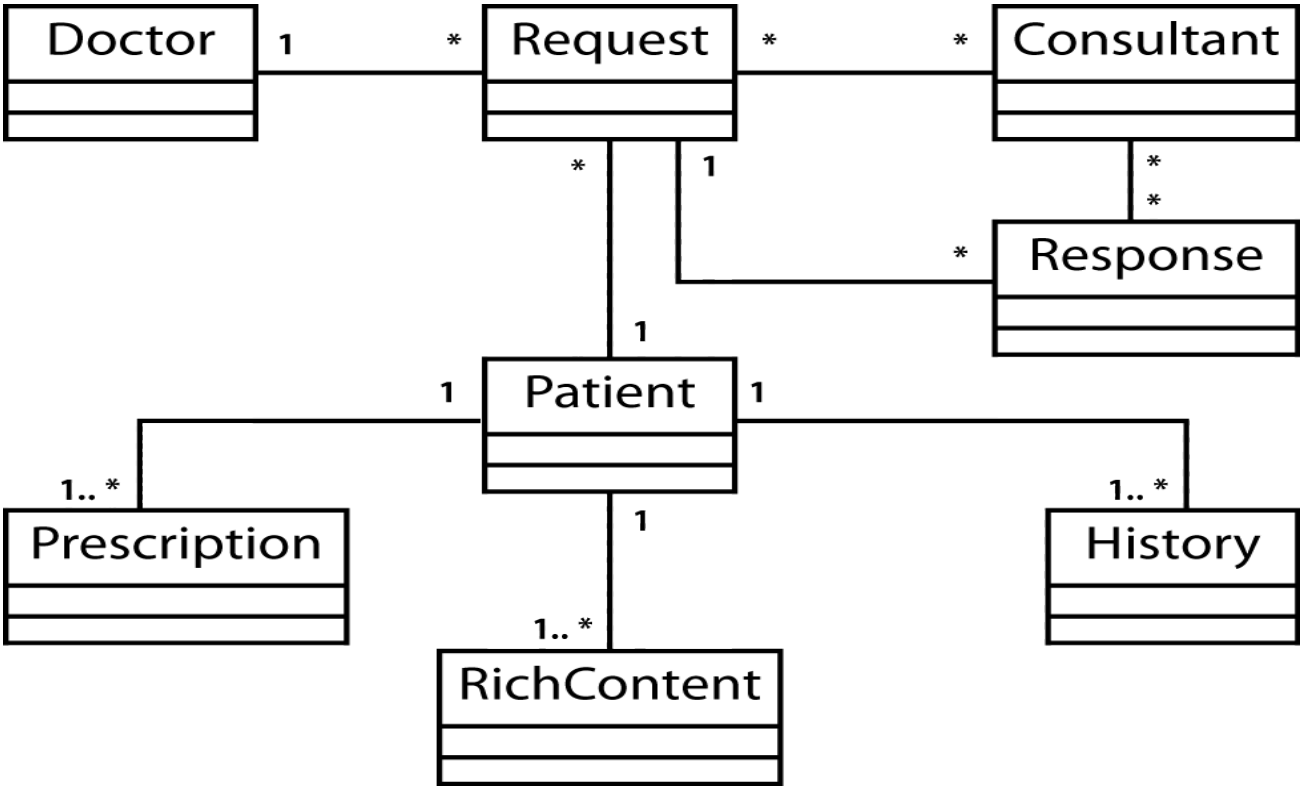

Illustration 1: Team Doctor application - Business Domain Model

2.2 Use Case Diagrams

The following use case diagram captures the main use cases that must be supported by your proposed design and implementation.

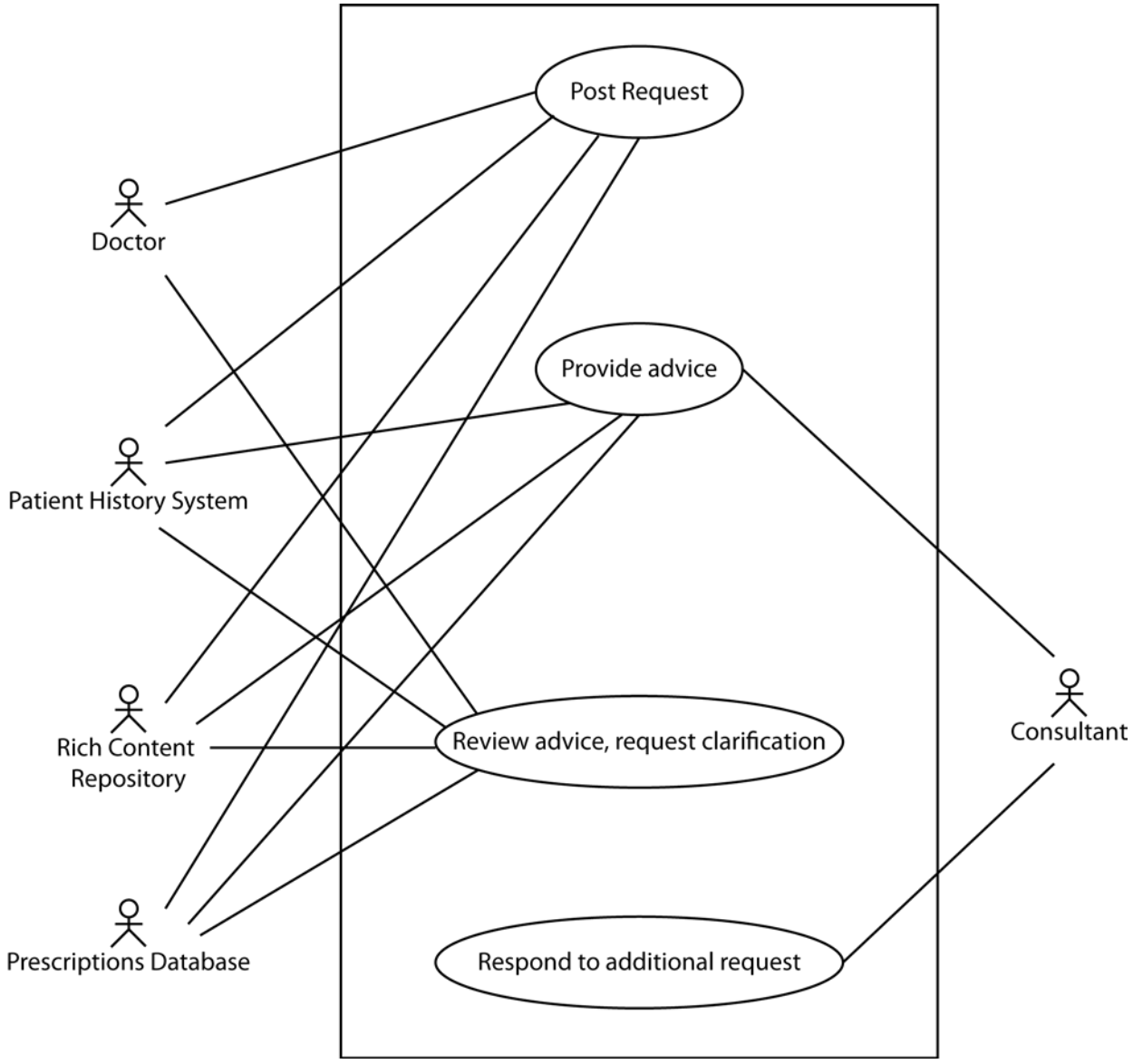

Illustration 2: Team Doctor application – Use Case diagram

The use case specifications provided are a first pass at the use cases and provide enough detail in order for you to architect a solution.

Use Case Specification — Post Request

Brief Description

The Post Request use case allows a doctor to post a request for information to the system so that it can be viewed and responded to by other qualified doctors and consultants.

Basic Flow

1. The doctor creates a new request for information providing additional information if necessary in the form of notes, or by attaching rich content, patient history, and previous prescription history. The doctor submits the request to the application.
2. The application validates and approves the request and publishes it for response to a set of qualified personnel registered in the application by placing the request in their work queue.

Use Case Specification — Provide Advice

Brief Description

The Provide Advice use case allows a recognized subject matter expert (doctor or consultant) to provide advice to a request for assistance.

Basic Flow

1. The subject matter expert (user) logs in to the system and reviews the first available request for assistance in his or her work queue.
2. The user reviews the request along with all supporting documentation and attachments and provides some initial advice before submitting the updated request to the application.
3. The application saves the submitted response and notifies the original requester that a response has been received by generating an update message and placing it in the requester's work queue.

Use Case Specification — Review Advice, Request Clarification

Brief Description

The Review Advice, Request Clarification use case allows the original requester to review advice provided by a subject matter expert and optionally, to request further clarification on it if necessary.

Basic Flow

1. The original requester (the user) logs in to the system and selects an open request from his or her work queue in order to review responses received. The system returns the request along with all responses received from subject matter experts.
2. The user reviews the responses along with all supporting documentation and attachments and selects one specific response to probe further on.
3. The user constructs a more detailed query, attaching any pertinent information if necessary, and submits the request to the application.
4. The application saves the submitted response and notifies the original responder that a clarification request has been received by generating an update message and placing it in the responder's work queue.

Use Case Specification — Respond to Additional Request

Brief Description

The Respond to Additional Request use case allows the original responder to review a clarification request received and to respond to that request, providing additional information and justification where appropriate for the original advice.

Basic Flow

1. The original responder (the user) logs in to the system and selects an open request from his or her work queue. The system returns the request along with all responses received from subject matter experts. The user reviews the clarification request along with all attached supporting documentation and attachments.
2. The user constructs a more detailed response, attaching any pertinent information if necessary, and submits the response to the application.
3. The application saves the submitted response and notifies the original requestor that a clarification response has been received by generating an update message and placing it in the requester's work queue.

3 Deliverables

It is your task to create an architecture and design for the System under Development (SuD) with the given business domain model, information provided above and requirements in the use cases. The architecture must be based on the Java Enterprise Edition (Java EE) platform[1]. All deliverables will be accepted as HTML documents only and each diagram must be UML compliant.

1. Create a class diagram for the SuD. Public method names referenced in other UML diagrams (for example, sequence diagrams) should be provided.
2. Create a Component diagram for the SuD showing the components used in the system and their interaction. Examples of components are Enterprise JavaBean (EJB), servlets, JavaServer Pages (JSP), major Plain Old Java Objects (POJOs), and important Managers / Controllers / Design Pattern implementations.
3. Create a Deployment diagram that describes the proposed physical layout of the major tiers of the SuD.
4. Create either a Sequence or Collaboration diagram for each use case provided.
5. List the top three technical risks you have identified in the project and identify a mitigation strategy for each risk.
6. Any assumptions made, during the process of creating the architecture and design that have a material impact on your design and any decisions made in arriving at that design, must be listed.

Your architecture and design will be graded on how well it supports the requirements detailed in this document and on the clarity of all information provided in both textual and diagrammatic form.

3.1 Submitting Your Work

Note to the candidate: Failure to follow the rules described here will result in an immediate failure and require a resubmission on your part.

1 Although we do not explicitly require Java EE 6, we expect J2EE 1.4 or higher to be used.

When you have completed your solution, you should have an "index.html" that has your name, testing ID, a link to the class, component, deployment diagrams, risk list, this instruction file, and a link to each of the sequence/collaboration diagrams. Build a JAR file that contains all HTML files. You must build a JAR file; do not send individual files.

The name of your submission JAR file MUST be derived from your testing ID. Your archive file name MUST BE in the format: `scea-<AAAAAAAAA>.jar`, where *AAAAAAAAA* is your testing ID.

4 Marking

The marking for this assignment is described below. You will receive a single score for parts 2 and 3.

Below are the sections of the assignment and points available per section with a minimum score per section if it is a required section to pass. The minimum score to pass the exam, if passing required sections, is 114. If a person fails any of the required sections Component, Class, Deployment or Question, then it is an automatic failure regardless of the final score. We will only post a pass or fail. If you fail, we will let you know which sections need improvement in order to pass so you do not have to focus on all sections with your resubmission.

	Points	Minimum Pass
Class Diagram	40	26
Component Diagram	40	26
Deployment Diagram	24	17
Interaction Diagrams	16	0
Risks & Mitigation List	16	0
Part 3 - Short Answers	24	17
Total	160	

