TOPCODER - ARCHITECTURE CHALLENGE SUBMISSION

# 1. Introduction

*ConnectCenter is a customer portal offering online claims and remittance management with Change Healthcare.A medical claims management solution for providers who want to improve claim accuracy, streamline submissions, and optimize denials-and-appeals management.*

## 1.1 Purpose

*A state-of-the-art solution for payers that want to outsource full or partial mailroom operations, including scanning, imaging and data capture, in order to streamline processes and focus on core business competencies.*

## 1.2 Scope

*The scope of the document is to reduce processing costs and Improve Accuracy and Productivity, Reduce Costs.*

* *Reduce human error by using state-of-the-art mailroom operations and industry leading scanning and data capture technology*
* *Unite mailroom functions with world-class scanning and imaging tools, decreasing manual intervention at the earliest point of entry into payers’ organizations.*
* *Automate mail sorting and scanning processes with Paper-to-EDI Conversion to dramatically improve accuracy and productivity, while at the same time reducing expenses.*
* *Get high-quality data and fewer pended claims as the result of data validation and editing tools that perform provider/member matching, eligibility verification and more.*
* *Retrieve documents fast and easy, as well as eliminate the need to maintain a paper claim inventory, with the Change Healthcare electronic image repository.*
* *Improve accuracy and productivity with fewer full-time employees, while significantly reducing administrative costs and capital investment in technology and equipment.*

## 1.3 Definitions, Acronyms and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the Architecture Document. List them as bullet points. See an example below]*

***Example:***

* UML: short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams.
* K8s: also known as Kubernetes, is an open-source system for automating deployment, scaling, and management of containerized applications.
* ...

## 1.3 References

*[This subsection provides a complete list of all documents referenced elsewhere in the Architecture Document. Identify each document by title, report number (if applicable), date, and publishing organization(if applicable). Specify the sources from which the references can be obtained.]*

*[The format should be]*

* *Journal/Article: <Title>, <Report Number (optional)>, <Date>, <Publishing Organization >*
* *Web page: <Title>, <URL>*

***Example:***

* Software-based control flow checking against transient faults in industrial environments, 2014, IEEE Trans Indust Inf
* HTTP/1.1 - RFC 2616, <https://www.ietf.org/rfc/rfc2616.txt>
* ...

# 2. Architectural Representation

*[This section presents the architecture as a series of views; use case view, logical view, process view, deployment view and implementation view etc. Not every view is required, you can freely pick which views are needed based on the requirement. The view should be presented in UML or other formal diagrams.]*

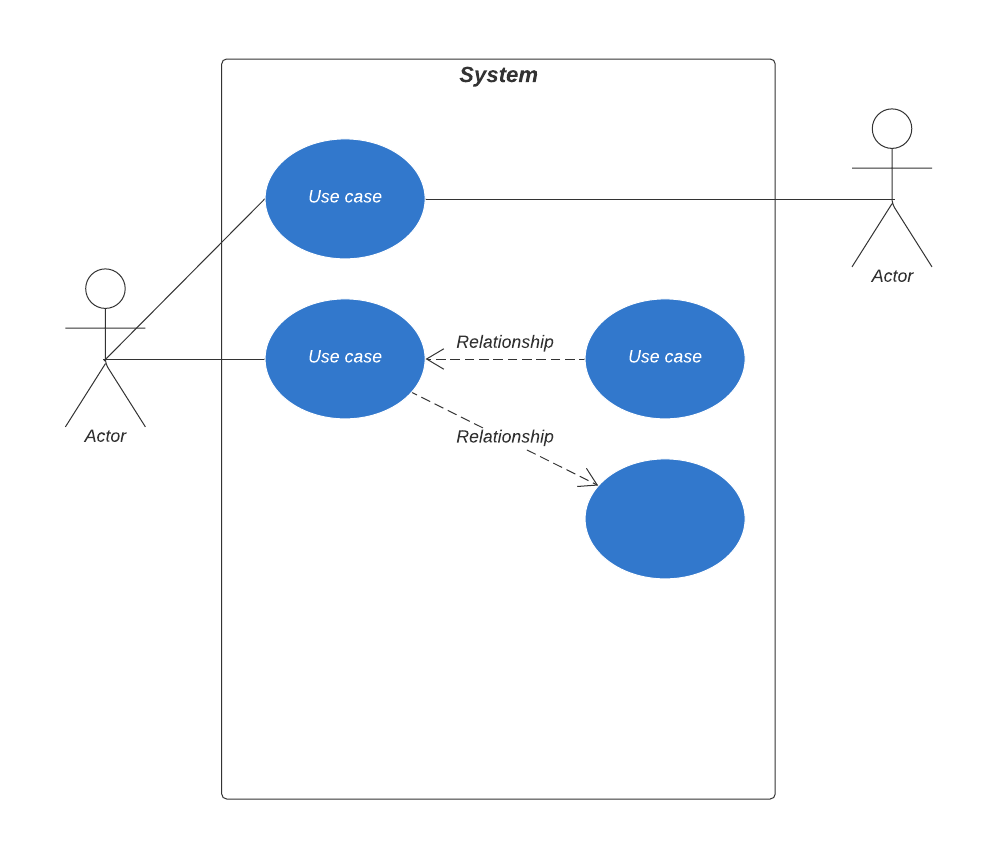
*[The recommended modeling tool is LucidChart: https://www.lucidchart.com/ ]*

## 2.1 Use Case View

*[A description of the use-case view of the software architecture. The Use Case View is an important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or*

*illustrate a specific, delicate point of the architecture.]*

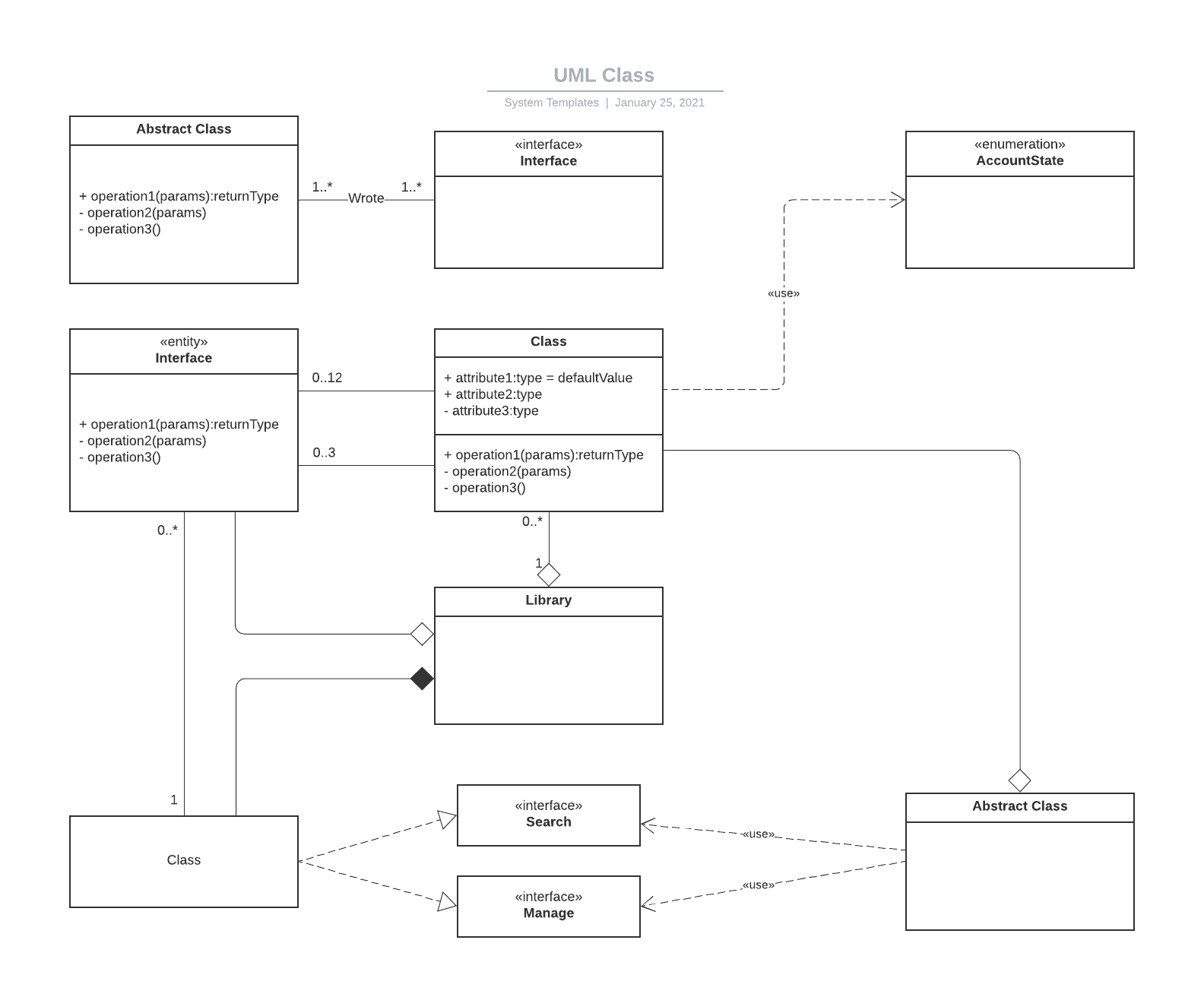
*[The Use Case Diagrams may be used to present the Use Case View.]*

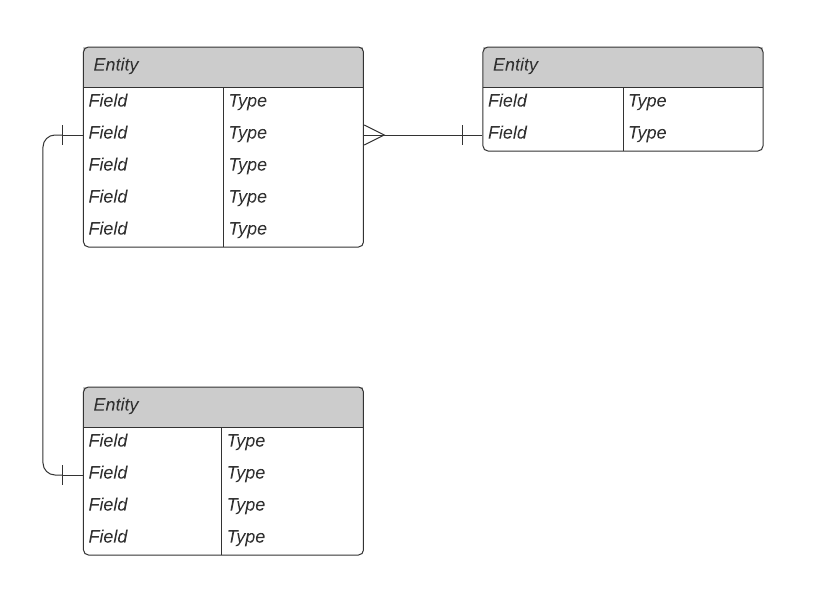
**

## 2.2 Logical View

*[The logical view of the architecture describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. It also describes the most important use-case realizations, for example, the dynamic aspects of the architecture.]*

*[Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers. If it contains Database, Database ER Diagrams might also be included.]*

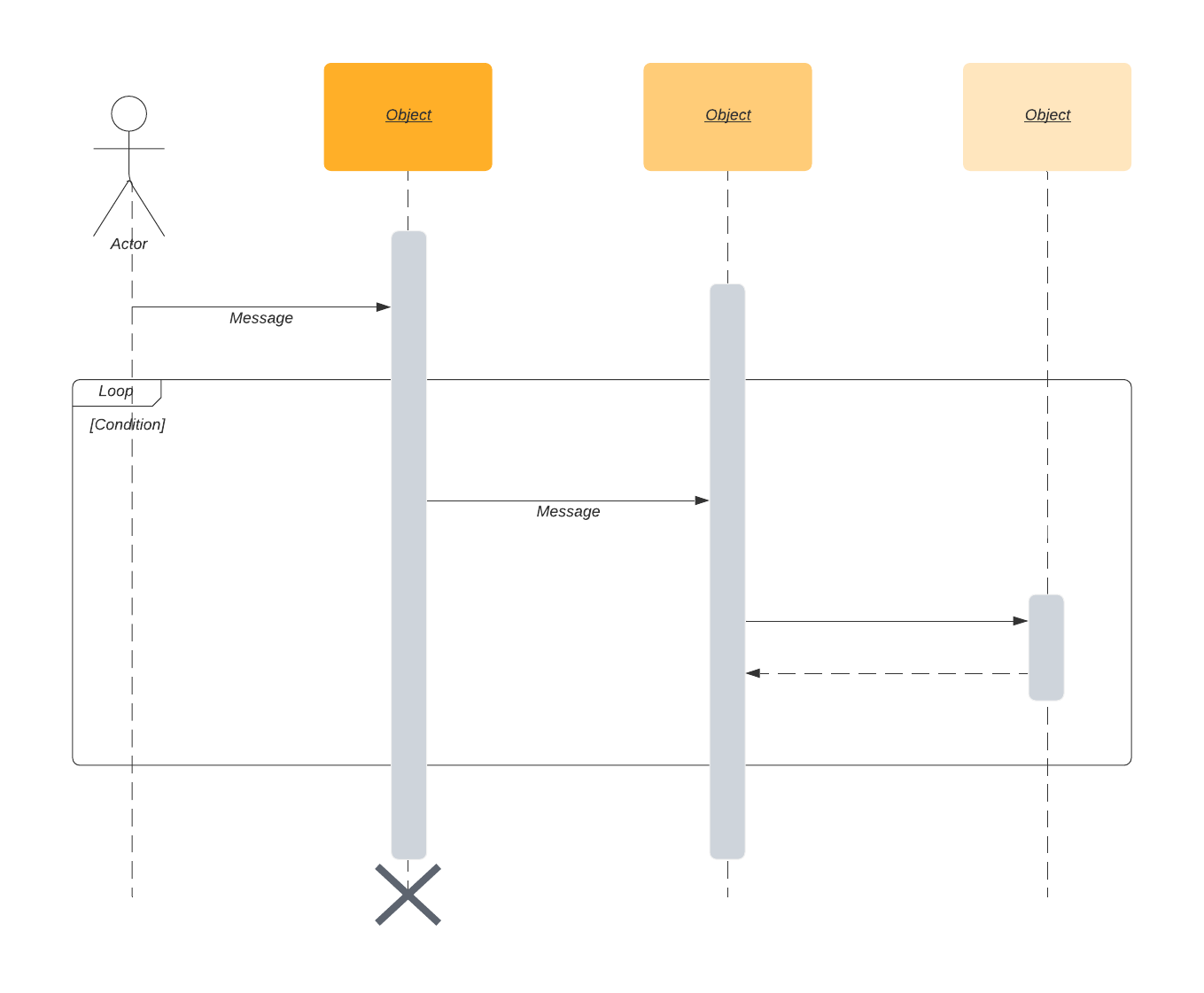
**

**

## 2.3 Process View

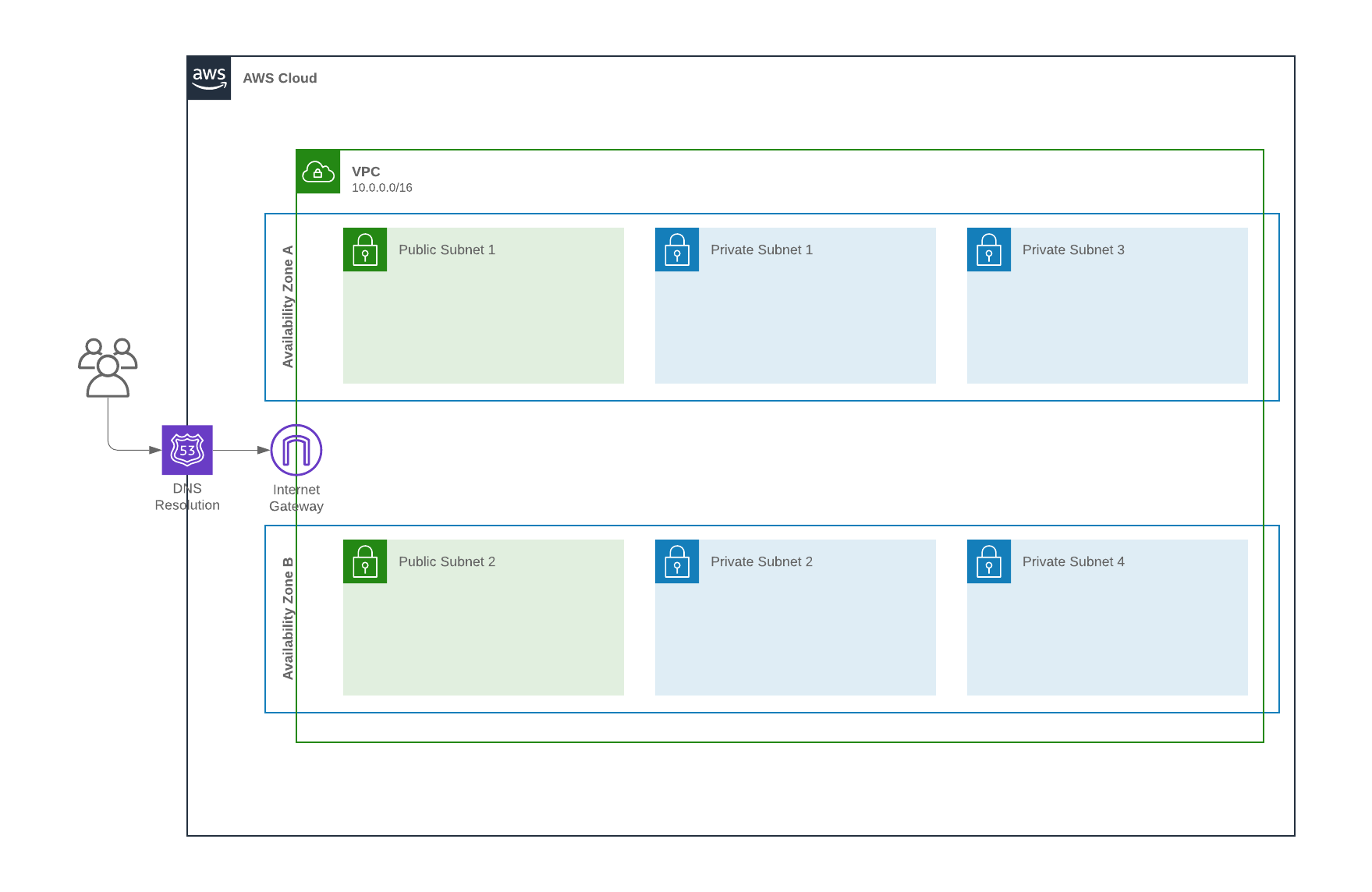
*[The process view describes the tasks (processes and/or threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.]*

*[Sequence diagrams/Class Diagrams/Flowchart may be included to illustrate the execution flow, and the interactions and configurations.]*



## 2.4 Deployment View

*[The deployment view describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes.]*



# 3. Size and Performance

*[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]*

***Example:***

The sizing and performance requirement is

* The system shall support up to 2000 simultaneous users against the central database at any given time, and up to 500 simultaneous users against the local servers at any one time.
* The system shall provide access to the legacy course catalog database with no more than a 10 second latency.
* The system must be able to complete 80% of all transactions within 2 minutes.
* The client portion shall require less than 20 GB disk space and 8 GB RAM.

To meet the requirement, the architecture supports the sizing and timing requirements through the implementation of a client-server architecture. The client portion is implemented on local campus PCs or remote dial up PCs. The components have been designed to ensure that minimal disk and memory requirements are needed on the PC client portion.

# 4. Quality

*[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: including but not limited to extensibility, reliability, portability, performance and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated as well.]*

***Example:***

The quality requirement is

* The mean time between failure (MTBF) should be less than 10 minutes.
* The data accuracy and consistency should be larger than 99.9999%

The architecture uses hot-packup to ensure the less MTBF, and Raft algorithm to ensure the high consistency.