Dumitru Alexandru David
1104

Design of Cloud-Based Information Systems: A Telemedicine Case Study

Contents

1. Introduction

The healthcare industry is currently undergoing a massive paradigm shift, transitioning from fragmented, on-premise legacy systems to integrated, cloud-based environments. Within this transforming landscape, this essay explores the design of a Cloud-Based Telemedicine and Remote Patient Monitoring System. This system is expressly designed to democratize access to healthcare by facilitating remote consultations and enabling the real-time monitoring of patient vitals through integrated IoT devices. By leveraging modern cloud capabilities, the proposed solution aims to bridge the physical gap between providers and patients, ensuring continuous care delivery regardless of geographical limitations. (Kanerika Inc, n.d.)

The primary objective of this essay is to rigorously detail the architectural design of this system, specifically justifying the adoption of a microservices architecture to ensure scalability and resilience. To bridge the gap between abstract requirements and concrete implementation, the essay demonstrates the system's structure and behavior through standard modelling languages. This includes the use of Unified Modeling Language (UML) to define the software blueprints and data structures, as well as Business Process Model and Notation (BPMN) to visualize and optimize the underlying clinical workflows.

2. History: From Mainframes to the Cloud

The evolution of healthcare information systems (HIS) mirrors the broader history of computing, characterized by a transition from centralized mainframes to distributed cloud architectures.

In the 1960s and 1970s, the concept of computing in healthcare was dominated by mainframes. These massive machines were installed in single departments, often without any connectivity to outside networks. They served purely administrative functions, such as basic billing and census management, with no real-time clinical utility. (Rice, n.d.)

The 1980s and 1990s saw the rise of the client-server model. Hospitals began implementing departmental systems (ex: Radiology Information Systems) hosted on local servers. While this improved internal efficiency, it created "data silos" where information could

not easily flow between departments or institutions (al., n.d.). The hardware required to run these systems was expensive to procure (Capital Expenditure) and difficult to maintain.

The 2000s and 2010s marked the emergence of virtualization and the early cloud. Amazon Web Services (AWS) launched its Elastic Compute Cloud (EC2) in 2006, fundamentally changing the economics of IT infrastructure (IBM, n.d.). However, healthcare was slow to adopt public cloud services due to strict regulatory concerns regarding patient data privacy (HIPAA).

Today, the landscape has shifted to Cloud-Native architectures. Modern systems utilize microservices—small, independent software components—that run in containers on cloud platforms. This allows telemedicine providers to scale their services instantly during high-demand periods (like the COVID-19 pandemic) without purchasing new hardware (Today, n.d.). This essay focuses on this modern era, modeling a system that leverages these cloud capabilities.

3. Modelling as the Language of Architecture

Software architecture is abstract; it deals with structures, interfaces, and behaviors that are invisible to the naked eye. To bridge the gap between abstract architectural concepts and concrete implementation, we use standard modeling languages.

For a cloud-based system, modelling serves two critical functions:

1. **Blueprint for Microservices:** In a monolithic system, everything is in one place. In a cloud microservices architecture, functions are scattered across servers. We use Class Diagrams and Communication Diagrams to define how these scattered services share data and interact, ensuring that the system remains cohesive despite being distributed (Paradigm, n.d.).
2. **Workflow Visualization:** Cloud systems often automate complex human workflows. BPMN and Activity Diagrams are essential for mapping the patient's journey—from booking an appointment to receiving a prescription—ensuring that the software aligns with actual medical procedures (Dynamics, n.d.).

The following sections present the specific models designed for our Telemedicine system.

4. System Design and Models

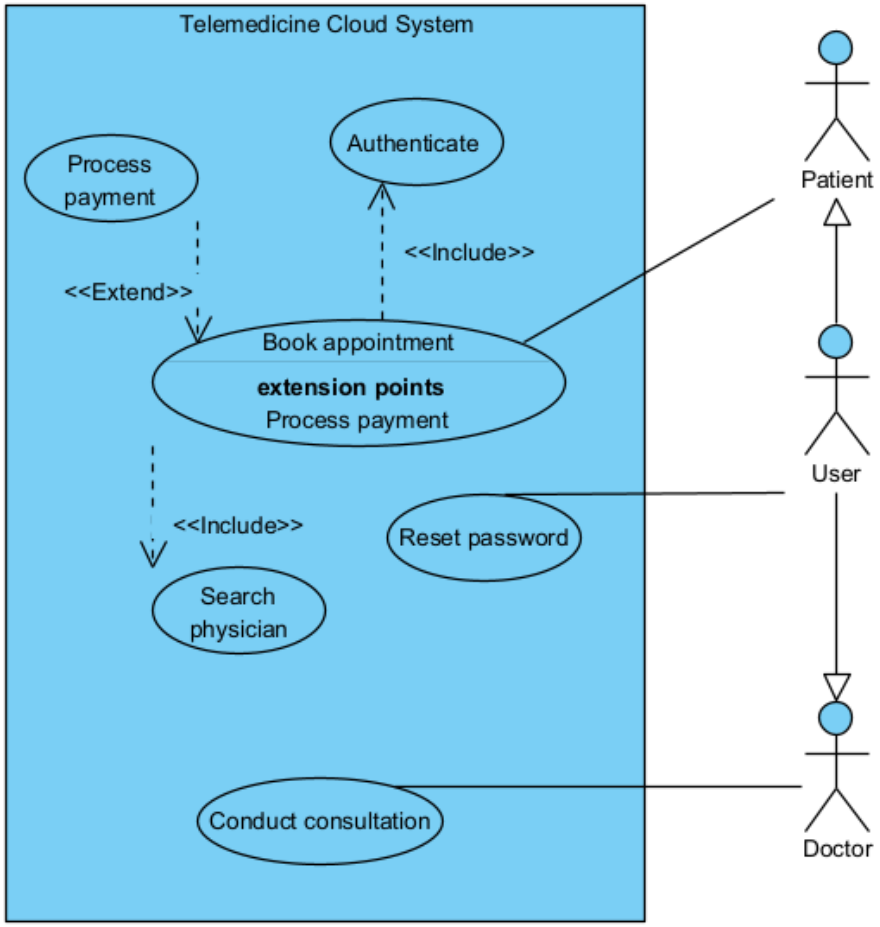
4.1 Use Case Diagram: Defining System Scope

The Use Case diagram defines the functional requirements of the system. It establishes the "System Boundary" and identifies the "Actors" (users) and external systems that interact with it.

Narrative: To capture the complexity of a modern cloud platform, we utilize Generalization and Dependencies. We introduce a base actor named "User," from which "Patient" and "Doctor" inherit; this avoids redundancy, as both must perform common actions like "Reset Password." (HL7, n.d.)

The core clinical flow uses advanced relationships:

1. <<include>>: The "Book Appointment" use case *includes* "Search Physician" because a patient cannot book without selecting a doctor first. Similarly, "Authenticate" is included in all secure transactions.
2. <<extend>>: The "Process Payment" use case extends "Book Appointment." This is a conditional flow—if the appointment is free or covered by insurance, the payment extension does not trigger.

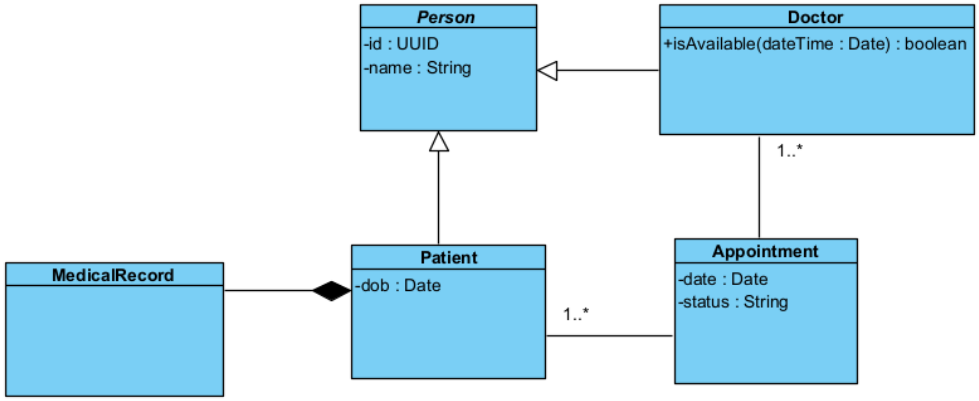


4.2 Class Diagram: The Data Model

The Class Diagram describes the static structure of the system, adhering to Object-Oriented principles to ensure scalability and data integrity.

Narrative: The system moves beyond simple data storage to a robust object model using Inheritance and Composition. Inheritance is when define an abstract class, Person, containing shared attributes (UUID, Name, Email). The Doctor and Patient classes inherit from Person, adding their specific attributes. Composition is when we model the Electronic Health Record (EHR) using a "Composition" relationship (a filled diamond). The Patient class is composed of MedicalRecord. This implies a strong lifecycle dependency: if a Patient is deleted from the

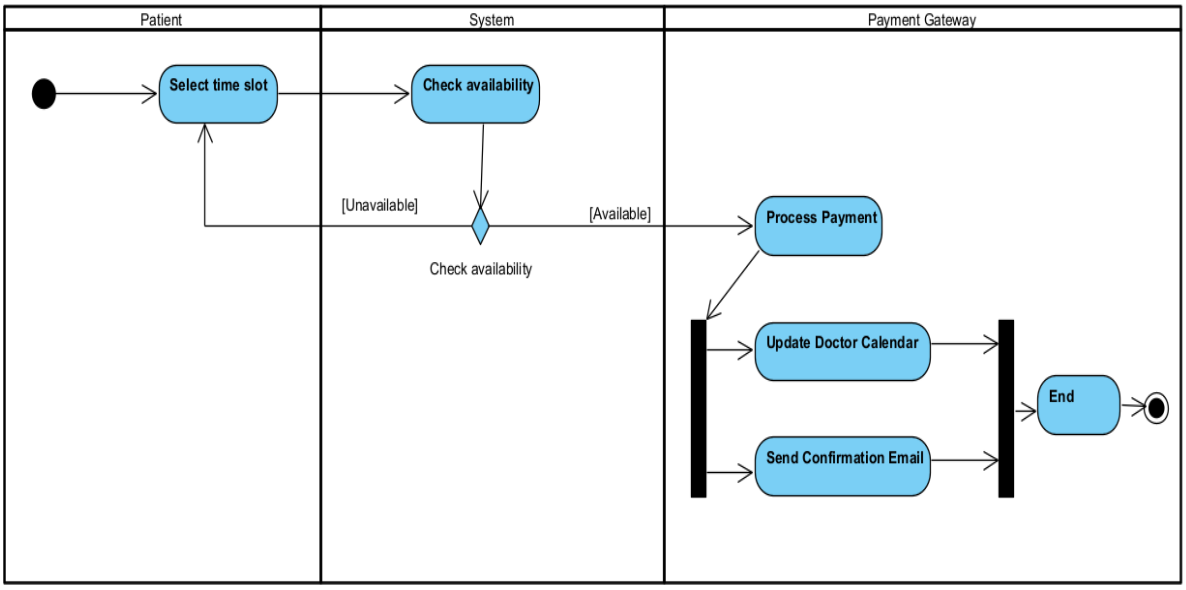
system, their Medical Record part cannot exist independently—it is destroyed or archived with the parent object. (HL7, n.d.)



4.3 Activity Diagram: The Appointment Workflow

The Activity Diagram models the dynamic flow of a specific procedure. Here, we model the Appointment Booking Flow.

Narrative: The process begins when the Patient selects a desired time slot. The system first checks the Doctor Schedule Database for availability. A Decision Node determines the path: if the slot is free, the flow proceeds to Payment Processing; if the slot is taken, the flow loops back to Select Time Slot. Once payment is confirmed, the system executes two parallel tasks (using a Fork Node): sending a confirmation email to the patient and updating the doctor's calendar.

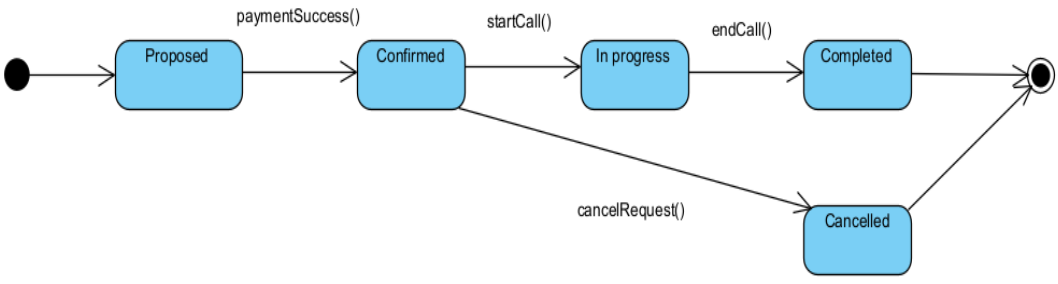


4.4 State Machine Diagram: Appointment Lifecycle

While the activity diagram shows the workflow, the State Machine diagram tracks the status of a single entity, which is the Appointment Object, throughout its life.

Narrative:

An appointment does not simply exist; it transitions through valid states. It begins in the Proposed state. Upon payment, it transitions to Confirmed. From there, it can transition to In-Progress (when the call starts) or Cancelled (if the user cancels). It cannot jump from Cancelled to In-Progress. This diagram is essential for developers to program the correct logic constraints in the backend services.

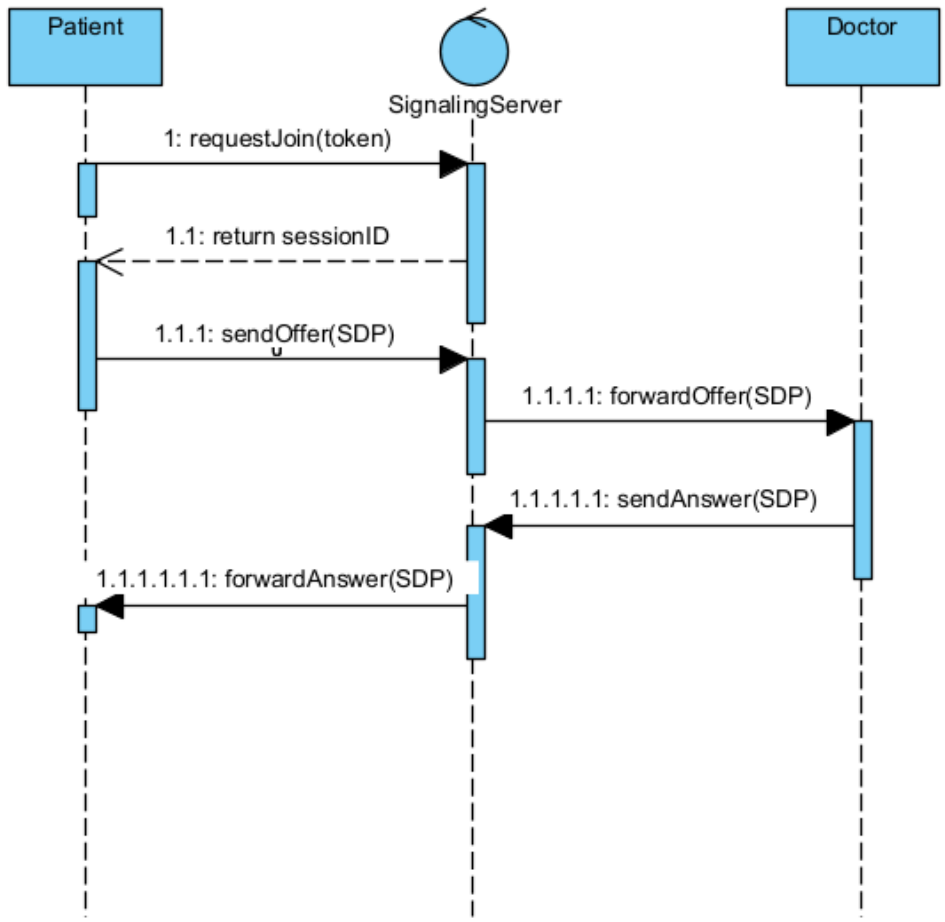


4.5 Interactions (Sequence) Diagram: The Video Consultation

Sequence diagrams are vital for detailing the timing of messages between system components. This diagram illustrates the WebRTC Signaling process (Mozilla, n.d.).

Narrative:

To establish a secure video link, the Patient's browser (Client) and Doctor's browser must exchange handshake data before the video starts. The diagram shows the Client sending a JoinRequest to the SignalingServer. The server validates the token and returns a SessionID. The Client then sends an Offer (SDP), which the Server forwards to the Doctor. This sequence ensures that both parties agree on encryption keys before any video data is transmitted (Mozilla, n.d.).

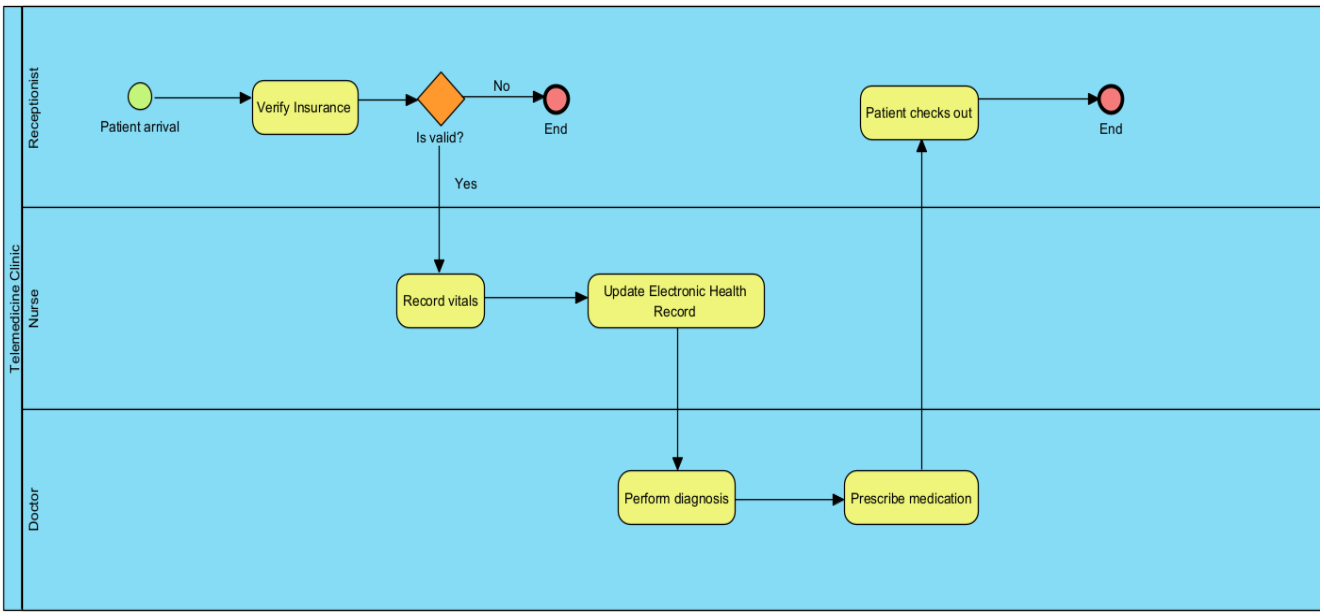


4.6 BPMN Diagram: Clinical Business Process

BPMN is used to communicate with non-technical stakeholders (hospital administrators).

Narrative:

The BPMN diagram illustrates the end-to-end Patient Triage Process. It uses a "Pool" to represent the Hospital, with "Lanes" for the Receptionist, Nurse, and Doctor. It visualizes the flow where a Receptionist verifies insurance, a Nurse records vitals, and the Doctor performs the diagnosis. Unlike UML, this diagram focuses on human responsibilities and hand-offs between departments.



5. Conclusion

The transition of healthcare information systems from the rigid, isolated mainframes of the 1960s to the agile, cloud-native environments of today represents more than a technological upgrade; it is a fundamental shift in how care is delivered. As explored in this essay, the move toward microservices and cloud infrastructure addresses the historical failures of "data silos" and capital-heavy hardware, offering a solution that is both scalable and resilient. However, the complexity inherent in these distributed systems necessitates a rigorous design approach. The architectural blueprint presented here demonstrates that successful telemedicine platforms rely not just on code, but on a comprehensive understanding of structure and behavior before a single line of software is written.

Through the application of Unified Modeling Language (UML) and Business Process Model and Notation (BPMN), we have bridged the gap between abstract requirements and concrete technical implementation. The use of Class and Use Case diagrams established a robust static foundation, ensuring data integrity through inheritance and composition while clearly defining the system boundary. Simultaneously, the dynamic models (Activity, State Machine, and Sequence diagrams) addressed the intricacies of system behavior, from the logic of appointment scheduling to the cryptographic security of WebRTC signaling. Furthermore, the inclusion of BPMN ensured that the human element of healthcare was not lost, mapping technical processes directly to clinical workflows like patient triage.

Ultimately, the design of this Cloud-Based Telemedicine System proves that modern software architecture is a multidimensional discipline. By decoupling data through microservices and strictly defining object lifecycles, the proposed system is capable of handling the high-demand scenarios of modern healthcare, such as pandemics or remote monitoring needs. This architectural rigor ensures that the technology remains a reliable enabler of the primary goal: democratizing access to healthcare and ensuring that patient care is continuous, secure, and accessible regardless of physical location.

Bibliography

al., M. S. (n.d.). *Microservices in Healthcare*. Retrieved from Marutitech.com

Dynamics, C. D. (n.d.). *BPMN in Healthcare*. Retrieved from Cloverdynamics.com

HL7. (n.d.). *FHIR Patient Resource*. Retrieved from Build.fhir.org

IBM. (n.d.). *History of Cloud Computing*. Retrieved from IBM.com

Kanerika Inc. (n.d.). *Medium*. Retrieved from https://medium.com/@kanerika/data-migration-in-healthcare-a-complete-guide-25d39f3dbeea

Mozilla. (n.d.). *WebRTC Signaling*. Retrieved from Developer.mozilla.org

Paradigm, V. (n.d.). *UML vs. Microservices*. Retrieved from Visual-paradigm.com

Rice, I. (n.d.). *Timeline of health software*. Retrieved from Timelines.issarice.com

Today, T. (n.d.). *Telemedicine Architecture*. Retrieved from Telehealthandmedicinetoday.com