Identity Based Health Care System

Ahsan Yasir Sharar 011201446
Kawsar Newaz Chowdhury 011201454
Md Tarifuzzaman 011191245
Susmita Dey Reia 011191188

Introduction:

Our mother project is ONE Nation ONE Identity, and we are creating a proper solution where everyone is traceable.

Our motivation to work on this project is that in Bangladesh, patients don't have their own database for their personal medical data. As patient histories are not kept in our country, we are trying to solve these complications. We wish to construct a patient database utilizing a unique identification because a crucial event in a patient's medical history is so significant. By employing the BM&DC verification number, we are building a doctor's database at the same time to ensure that all registered doctors in the hospital are legitimate. The doctor verification policy, doctor education, degree, and other information are all included in this database. We'll also keep tabs on hospitals that aren't registered. The hospital ID will also be used to verify hospitals so that patients can confidently choose the hospitals and doctors who specialize in their condition.

Tactics-Based Questionnaire for Availability

Tactics Group	**Tactics Question**	**Support (Y/N)**	**Risk**	**Design Decision & Location**	**Rationale & Assumptions**
Detects Faults	Does the system use faults ping/echo to detect failure of a component or connection, or network congestion?	Y	The system may generate false positives, which can lead to unnecessary downtime or performance degradation	This tactic is typically implemented in the system's monitoring and alerting infrastructure.	This tactic is effective for detecting failures of components and connections that are actively communicating.
Detects Faults	Does the system use a monitoring system to detect failure or congestion in the network?	Y	If the monitoring system fails, the system may not be able to detect network failures or congestion in a timely manner	The system is designed to monitor all critical network components and identify any signs of failure or congestion	The system is able to collect and analyze data from the network.
Detects Faults	Does the system detect incorrect sequences of events?	N	If the system is unable to detect incorrect sequences of events, it may not be able to respond appropriately to errors		
Detects Faults	Does the system compare computational results from multiple sources	Y	Data Inconsistency	Implementing a Data Fusion Module that compares results from various health data sources	The assumption is that comparing data from multiple sources will enhance the accuracy of health information.

Tactics Group	**Tactics Question**	**Support (Y/N)**	**Risk**	**Design Decision & Location**	**Rationale & Assumptions**
Detects Faults	Does the system alter the normal flow of execution?	N	Data Integrity and System Stability		
Recover from Faults	Does the system refer to a configuration in which one or more duplicate components can step in and take over the work if the primary component fails?	Y	Downtime and Data Loss	Implementing Redundant Components for High Availability	Assumes that minimizing downtime and potential data loss is crucial for the reliability of the identity based health system.
Recover from Faults	Does the system refer to operating a previously failed or in-service upgraded component in a "shadow mode" for a predefined duration of time prior to reverting the component back to an active role?	N	Extended Downtime and Data Inconsistency		
Recover From Faults	Does the system assume that the fault that caused a failure is transient, and that retrying the operation may lead to success?	Y	Increased Load on the System	Implementing Retry Mechanism in the Component Handling Failures	Assumes that transient faults are common, and retrying operations may lead to successful execution, mitigating the impact of transient faults.

Tactics Group	**Tactics Question**	**Support (Y/N)**	**Risk**	**Design Decision & Location**	**Rationale & Assumptions**
Prevent Faults	Does the system refer to temporarily placing a system component in an out-of-service state for the purpose of mitigating potential system failures?	Y	Temporary Service Disruption Stability	Temporarily Isolate Component for Maintenance	Assumes that temporarily isolating a component for maintenance can prevent potential system failures and improve overall system stability
Prevent Faults	Does the system targeting high availability services leverage transnational semantics to ensure that asynchronous messages exchanged between distributed components are "ACID properties"?	Y	Potential Overhead on System Performance	Implementing Transnational Semantics for Asynchronous Messages	Assumes that maintaining "ACID properties" in the exchange of asynchronous messages contributes to the reliability and consistency of the identity-based health system.
Prevent Faults	Does the system ensure that the system is operating within its nominal operating parameters and take corrective action when the system nears a critical threshold?	N	Potential for System Instability		

Tactics-Based Questionnaire for Deployability

Manage deployment pipeline

Tactics Group	**Tactics Question**	**Support (Y/N)**	**Risk**	**Design Decision & Location**	**Rationale & Assumptions**
Scale roll outs	Is the purpose of scaled rollouts to deploy a new version of a service gradually to controlled subsets of the user population, allowing for monitoring, measurement, and the possibility of rollback to minimize potential negative impacts of deploying a flawed service?	Y	Limited Impact on Entire User Base	Implementing a phased scale rollout strategy. Embedded within deployment and release management processes	Assumes that a scaled rollout strategy is beneficial for minimizing the impact of deploying a flawed service in the identity-based health system.
Roll Back	Is there a mechanism in place to rollback a deployment to its prior state if defects are discovered or if it does not meet user expectations, considering that deployments may involve multiple coordinated updates of multiple services and their data, and the rollback mechanism must keep track of all these updates?	N	The risk of data inconsistencies, incomplete restoration, and challenges in handling complex system configurations, potentially compromising system stability during deployment.		
Script deployment commands	Are deployment scripts treated like code in the development process, involving documentation, code review, testing, and version control to ensure precision and reliability?	Y	Errors, misconfigurations, and security vulnerabilities, potentially leading to deployment failures or system instability.	Adopting microservices architecture for system flexibility. Location: Embedded at the architectural level of the system.	documentation, code review, testing, and version control, is crucial for preventing script misconfigurations and errors in the identity-based health system.

Manage Deployed System

Tactics Group	Tactics Question	Support (Y/N)	Risk	Design Decision & Location	Rationale & Assumptions
Manage service interactions	Can multiple requests from a client be directed to either version of the system services in any sequence?	Y	Service disruptions, compatibility issues, and potential system instability during deployment.	Implementing Simultaneous Deployment of Multiple Versions	Assumes that accommodating simultaneous deployment of multiple versions enhances system flexibility and reliability, mitigating version compatibility issues in the identity-based health system.
Package dependencies	Are elements packaged together with their dependencies to ensure they are deployed together?	N	Potential Dependency Inconsistencies Mechanism		
Feature toggle	Is there a "kill switch" or feature toggle integrated into the system for new features, allowing automatic disabling of features at runtime without requiring a new deployment?	Y	Unforeseen Issues in New Features	To enhance the system's flexibility and allow for seamless control over features at runtime, a feature toggle will be integrated into the system. This feature toggle will act as a "kill switch," enabling the automatic disabling of specific features without the need for a new deployment.	Assumes that integrating a "kill switch" or feature toggle is beneficial for effectively managing issues in new features without necessitating a new deployment in the context of the identity-based health system.

Utility Tree

Utility Attribute	**Attribute Refinement**	**ASR Scenario**
Performance	Transaction response time	While the system is under peak load, the transaction to update a patient's account in response to a change-of-address notification is completed in less than 0.85 seconds by the user.
	Throughput	The system can complete 130 normalized transactions per second at peak load
	Scalibility	Without a significant decrease in response time, the system is capable of accommodating a 50 percent increase in the number of simultaneous user interactions.
Usability	Proficiency training	With just one week of training, a new employee possessing two or more years of experience in the business can acquire the ability to execute any of the system's core functions in less than 7 seconds.
	Efficiency of operations	While interacting with a patient, a hospital payment officer initiates a payment plan and successfully completes the process without any input errors.
	Intuitive user interface	A patient without any prior experience in using computers, particularly an elderly individual, can effortlessly navigate the system to access their medical records.
Conductibility	Data conductibility	The configuration team successfully makes and tests the change to increase the fee for a particular service within one working day, and no alterations to the source code are necessary
	Penalization	Personalized content based on each patient's medical condition and preferences can be displayed by tailoring the system.
Maintainability	Routine changes	Encountering a response-time deficiency, a maintainer addresses the bug, and the bug fix is distributed with no more than 3 person-days of effort.
	Code readability	New developers can understand and modify the co debase within a week of on boarding.
	Version control	Version control is employed to manage the system's source code, facilitating seamless collaboration and tracking of changes.

Security	Confidentiality	A physical therapist has permission to access the section of a patient's record related to orthopedic treatment; however, they are not authorized to view other sections or any financial information. user.
	Resisting attacks.	The system repels an unauthorized intrusion attempt and reports the attempt to authorities within 90 seconds.
	Data encryption	Patient records and sensitive information undergo encryption both when stored and during transmission.
	Role-based access control	Only authorized personnel can access and modify patient records based on their roles and permissions.
Availability	No downtime	The new software from the database vendor is released and seamlessly replaced without causing any downtime.
	Continuous monitoring	The system is continuously monitored, and potential issues are proactively addressed to ensure maximum up time
	Disaster recovery	The system is capable of recovering from hardware failures or catastrophic events within a 4-hour time frame, guaranteeing minimal data loss.
	Redundancy	The system ensures availability in the event of failures by having redundant backups for critical components.
Reliability	Error handling	The system adeptly manages errors and furnishes users with informative error messages to aid in troubleshooting.
	Data integrity	Patient data is consistently accurate, and thorough checks are in place to prevent data corruption or loss
Interoperability	Integration with external systems	The healthcare system integrates seamlessly with external lab systems to receive test results and medical reports.
	Code readability	New developers can understand and modify the co debase within a week of on boarding.

Accessibility	Web accessibility	The system adheres to WCAG guidelines, ensuring accessibility for users with disabilities.
	Multilingual support	The system enables patients from diverse backgrounds to easily access their information by supporting multiple languages.
Privacy	Consent management	The system acquires explicit consent from patients before accessing or sharing any personal health information.
	Unionization	Patient data utilized for research or analysis undergoes unionization to safeguard their identities.

Pattern

For our Identity-Based Healthcare System we use Microservices Architecture Pattern.

Why Microservices Architecture?

For our Identity-Based Healthcare System, we use the Microservices Architecture Pattern. Among the architecture patterns, Microservices Architecture Pattern stands out as the most suitable choice for an identity-based healthcare system. This architecture offers several advantages that align well with the specific requirements of healthcare systems, including:

1. **Scalability:** Microservices architecture enables independent scaling of individual services, allowing the system to effectively handle increasing traffic and data volume. This is particularly crucial for healthcare systems that experience fluctuating demand and need to accommodate spikes in usage.
2. **Flexibility:** Microservices are independently deployable and updatable, facilitating seamless adaptation to changing requirements and evolving healthcare practices. This flexibility is essential for healthcare systems that need to stay abreast of advancements in medical treatments and technologies.
3. **Fault Isolation:** The microservices approach isolates faults within individual services, preventing them from cascading across the entire system. This compartmentalization enhances system resilience and ensures that service disruptions don't lead to complete system outages.
4. **Ease of Development:** Microservices are inherently smaller, more focused, and have well-defined boundaries, making them easier to develop and reduce the burden on healthcare organizations.

Considering the specific features of an identity-based healthcare system, microservices architecture proves to be a strong fit:

- **Sign Up:** Microservices can handle user registration and authentication efficiently, ensuring secure access to the healthcare platform.
- **Doctor & Patient Community:** Microservices can facilitate seamless interactions between doctors and patients, enabling online consultations, virtual appointments, and secure communication channels.
- **Search Question:** Microservices can power search functionalities, allowing users to quickly find relevant medical information, healthcare resources, and answers to their health-related queries.
- **Doctor Dashboard:** Microservices can equip doctors with a centralized dashboard to manage patient appointments, review medical records, communicate with patients, and access clinical decision support tools.

- **Patient History:** Microservices can securely store and manage patient medical histories, enabling comprehensive healthcare management and informed decision-making.
- **Hospital Info:** Microservices can provide easy access to hospital information, including location, services offered, contact details, and insurance plans accepted.
- **Message/Video:** Microservices can facilitate secure and HIPAA-compliant messaging and video conferencing between doctors and patients for consultations, follow-ups, and remote patient monitoring.

Doctor's Part

1. **Find Patient:** Microservices can enable efficient patient search and patient record retrieval for doctors.
2. **Mobile Banking:** Microservices can integrate with secure payment gateways to facilitate online bill payments and manage financial transactions.
3. **Add Prescription:** Microservices can streamline prescription management, allowing doctors to electronically prescribe medications and manage patient medication lists.
4. **Add Drug:** Microservices can provide access to comprehensive drug information and databases for informed prescription decisions.
5. **Get Appointment:** Microservices can manage appointment scheduling, enabling patients to book appointments online, track appointment status, and receive reminders.
6. **Payment Status:** Microservices can provide real-time payment status updates for patients and healthcare providers, ensuring transparency and timely payment processing.

Overall, the microservices architecture pattern offers a compelling solution for identity-based healthcare systems, enabling scalability, flexibility, fault isolation, and ease of development. Its alignment with the specific features of an identity-based healthcare system makes it an ideal choice for modernizing and streamlining healthcare delivery.

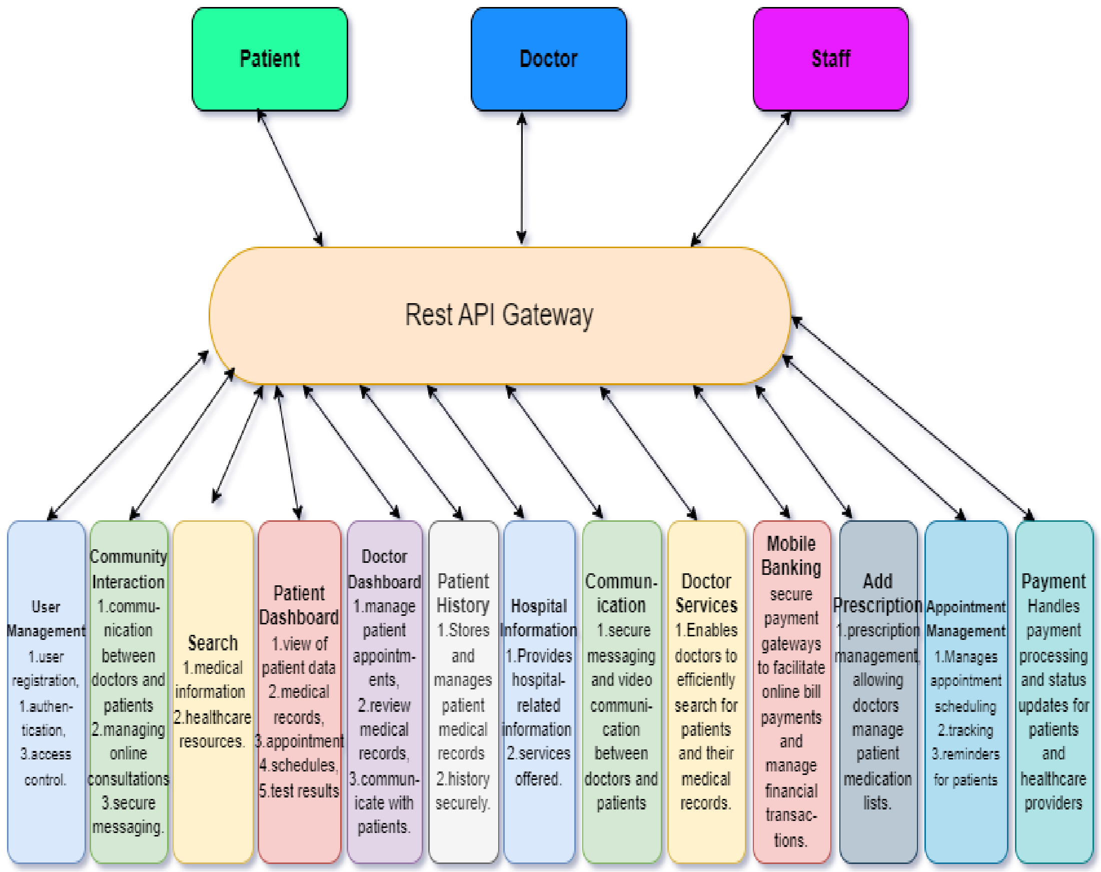

Patient
Doctor
Staff
Rest API Gateway
User Management 1.user registration, 1.authen-tication, 3.access control.
Community Interaction 1.commu-nication between doctors and patients 2.managing online consultations 3.secure messaging.
Search 1.medical information 2.healthcare resources.
Patient Dashboard 1.view of patient data 2.medical records, 3.appointment 4.schedules, 5.test results
Doctor Dashboard 1.manage patient appointm-ents, 2.review medical records, 3.commun-icate with patients.
Patient History 1.Stores and manages patient medical records 2.history securely.
Hospital Information 1.Provides hospital-related information 2.services offered.
Commun-ication 1.secure messaging and video communi-cation between doctors and patients
Doctor Services 1.Enables doctors to efficiently search for patients and their medical records.
Mobile Banking secure payment gateways to facilitate online bill payments and manage financial transac-tions.
Add Prescription 1.prescription management, allowing doctors manage patient medication lists.
Appointment Management 1.Manages appointment scheduling 2.tracking 3.reminders for patients
Payment Handles payment processing and status updates for patients and healthcare providers

Figure 1: Microservice Architecture Diagram

Comparison

Feature	**Microservices Architecture Pattern**	**Layered Architecture**	**Event-Driven Architecture**	**Microkernel Architecture Pattern**	**Space-Based Architecture**
Sign Up	Efficient user registration and authentication through dedicated microservices	Requires integration with the user management layer.	May involve real-time event processing for signup confirmation.	May require custom plugin development for user management.	May involve configuration management for signup functionality
Doctor & Patient Community	Facilitates seamless interactions through dedicated communication microservices	Requires integration with communication and user management layers.	May involve real-time event processing for community interactions	May require custom plugin development for community features.	May involve configuration management for community functionality
Search Question	Enables efficient search through dedicated search microservices.	Requires integration with search and data access layers.	May involve real-time event processing for search indexing.	May require custom plugin development for search functionality	May involve configuration management for search functionality
Patient Dashboard	Provides a personalized view through dedicated dashboard microservices.	Requires integration with data access and presentation layers.	May involve real-time event processing for dashboard updates.	May require custom plugin development for dashboard features.	May involve configuration management for dashboard functionality.
Doctor Dashboard	Empowers doctors with a centralized dashboard through dedicated microservices.	Requires integration with data access and presentation layers.	May involve real-time event processing for dashboard updates.	May require custom plugin development for dashboard features.	May involve configuration management for dashboard functionality.

Patient History	Stores and manages records securely through dedicated microservices	Requires integration with data access and security layers.	May involve real-time event processing for history updates.	May require custom plugin development for dashboard features.	May involve configuration management for patient history management
Hospital Info	Provides easy access to information through dedicated microservice	Requires integration with data access and presentation layers.	May involve real-time event processing for information updates.	May require custom plugin development for hospital information retrieval.	May involve configuration management for hospital information retrieval
Message or Video	Enables secure communication through dedicated communication microservices.	Requires integration with communication and security layers	May involve real-time event processing for message delivery.	May require custom plugin development for communication features.	May involve configuration management for communication functionality.
Doctor Part	Streamlines doctor tasks through dedicated microservices.	Requires integration with patient management, prescription management, and payment processing layers.	May involve real-time event processing for task updates	May require custom plugin development for doctor-specific features.	May involve configuration management for doctor-specific functionality.
Find Patient	Enables efficient patient search through dedicated patient management microservices.	Requires integration with patient management and search layers.	May involve real-time event processing for patient data updates.	May require custom plugin development for patient search functionality	May involve configuration management for patient search functionality

Mobile Banking	Integrates with secure payment gateways through dedicated payment processing microservices.	Requires integration with payment processing and security layers.	May involve real-time event processing for payment transactions	May require custom plugin development for mobile banking functionality	May involve configuration management for mobile banking functionality.
Add Prescription	Streamlines prescription management through dedicated prescription management microservices.	Requires integration with patient management, drug information, and security layers.	May involve real-time event processing for prescription updates.	May require custom plugin development for prescription management functionality.	May involve configuration management for prescription management functionality.
Add Drug	Provides access to comprehensive drug information through dedicated drug information microservices.	Requires integration with drug information and data access layers.	May involve real-time event processing for drug information updates.	May require custom plugin development for drug information retrieval	May involve configuration management for drug information retrieval.
Get Appointment	Manages appointment scheduling through dedicated appointment management microservices.	Requires integration with appointment management, patient management, and calendar layers.	May involve real-time event processing for appointment updates.	May require custom plugin development for appointment management functionality.	May involve configuration management for appointment management functionality.
Payment Status	Provides real-time payment status updates through dedicated payment processing microservice	Requires integration with payment processing and data access layers.	May involve real-time event processing for payment status updates.	May require custom plugin development for payment status retrieval	May involve configuration management for payment status retrieval.

Table 1: Additional Features Comparison

Attribute-Driven Design 3.0 for an Identity-Based Healthcare Solution

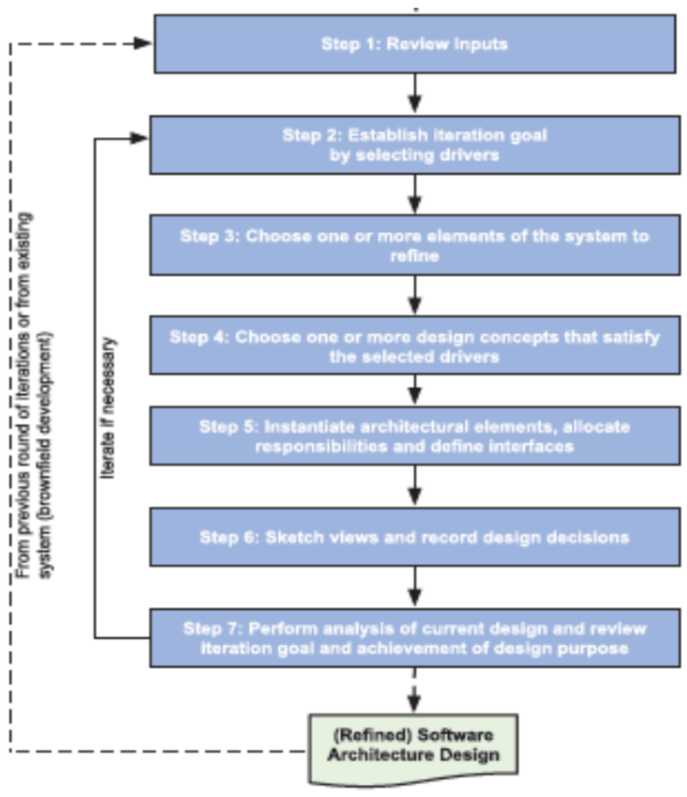

Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Perform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary

Figure 2: ADD Version 3.0

Iteration 1

- **Review Inputs:**
  - **Step 1:** Identify stakeholders: This includes patients, healthcare providers, administrators, and any other parties involved in the healthcare ecosystem.
  - **Step 2:** Gather user stories and requirements: Understand user needs, expectations, and pain points regarding identity management in the healthcare system.
  - **Step 3:** Analyze existing systems and data sources: This will include reviewing current patient identity systems, healthcare databases, and any other relevant IT infrastructure.
  - **Step 4:** Identify regulatory and compliance requirements: Ensure the identity solution adheres to all applicable healthcare regulations and data privacy laws.
- **Establish Iteration Goal by Selecting Drivers:**
  - **Step 1:** Identify Key Security Attributes:
    - Data confidentiality

    - Data integrity
    - Authentication and authorization
    - Access control
    - Non-repudiation
    - Audit trails
    - Threat and vulnerability management
    - Incident response
  - **Step 2:** Prioritize Security Pain Points:
    - Review user feedback, security audits, and vulnerability assessments to identify the most pressing security concerns.
    - Focus on areas that pose the highest risks to sensitive patient information or system integrity.
  - **Step 3:** Align with Security Regulations:
    - Ensure the chosen drivers comply with relevant healthcare data privacy laws and security standards, such as HIPAA, HITRUST, and NIST.
    - Consider any industry-specific or organizational security policies that need to be addressed.

Iteration 2

- **Establish Iteration Goal by Selecting Drivers:**
  - **Step 1:** Identify Key Privacy Attributes:
    - Data minimization: Collect and store only the minimum necessary personal information.
    - Purpose limitation: Use personal data only for the stated and authorized purposes.
    - Transparency: Provide clear and accessible information to users about data collection, use, and sharing practices.
    - Individual control: Empower users with control over their personal data, including access, correction, deletion, and sharing preferences.
    - Data retention: Establish clear policies for how long personal data is retained and securely dispose of it when no longer needed.
    - Data anonymization and pseudonymization: Consider techniques to deidentify or mask personal data when possible.
  - **Step 2:** Prioritize Privacy Pain Points:
    - Review user feedback, privacy audits, and risk assessments to identify the most pressing privacy concerns.
    - Focus on areas that pose the highest risks to patient privacy or data protection.
  - **Step 3:** Align with Privacy Regulations:
    - Ensure the chosen drivers comply with relevant privacy laws and regulations, such as HIPAA, GDPR, and CCPA.
    - Consider any industry-specific or organizational privacy policies that need to be addressed.
- **Choose One or More Elements of the System to Refine:**
  - **Step 1:** Focus on specific areas needing improvement: Identify components of the identity system that require optimization based on privacy.

  - **Step 2:** Start with high-impact areas: Prioritize refining components that offer the most significant improvements in privacy.
  - **Step 3:** Consider the overall system architecture: Ensure any changes to individual elements align with the overall design and do not introduce any unintended consequences.

Iteration 3

- **Establish Iteration Goal by Selecting Drivers:**
  - **Step 1:** Identify Key Interoperability Attributes:
    - Data exchange standards: Ensure compliance with common healthcare data exchange standards (e.g., HL7 FHIR, DICOM, SNOMED CT).
    - System integration: Facilitate seamless integration with other healthcare IT systems, including EHRs, PHRs, lab systems, imaging systems, etc.
    - API development: Create well-documented APIs to enable secure data exchange and communication between systems.
    - Workflow integration: Support coordinated workflows across different healthcare providers and organizations.
    - Data harmonization: Address semantic and syntactic differences in data formats and structures.
  - **Step 2:** Prioritize Interoperability Pain Points:
    - Review user feedback, system compatibility reports, and integration challenges to identify the most pressing interoperability issues.
    - Focus on areas that hinder data sharing and collaboration among healthcare providers.
  - **Step 3:** Align with Interoperability Initiatives:
    - Ensure the chosen drivers align with national or regional interoperability initiatives, such as ONC's Trusted Exchange Framework and Common Agreement (TEFCA) or EU's eHealth Digital Service Infrastructure (eHDSI).
    - Consider any industry-specific interoperability standards or requirements.
- **Choose One or More Elements of the System to Refine:**
  - **Step 1:** Focus on specific areas needing improvement: Identify components of the identity system that require optimization based on interoperability.
  - **Step 2:** Start with high-impact areas: Prioritize refining components that offer the most significant improvements in interoperability.
  - **Step 3:** Consider the overall system architecture: Ensure any changes to individual elements align with the overall design and do not introduce any unintended consequences.
- **Choose One or More Design Concepts That Satisfy the Selected Drivers:**
  - **Step 1:** Research and explore existing solutions: Evaluate existing identity management solutions in healthcare and other industries to identify potential best practices.
  - **Step 2:** Consider emerging technologies: Analyze the potential of new technologies, such as blockchain or biometrics, to address specific drivers.
  - **Step 3:** Develop new design concepts: When necessary, innovate and develop novel approaches to identity management tailored to healthcare needs.

- **Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces:**
  - **Step 1:** Implement the selected design concepts: Develop the technical specifications and components based on the chosen design concepts.
  - **Step 2:** Allocate responsibilities: Clearly define ownership and accountability for different aspects of the system design and implementation.
  - **Step 3:** Define clear interfaces: Establish standardized ways for different components of the identity system to communicate and interact with each other.

Iteration 4:

2. Establish Iteration Goal by Selecting Drivers:

- **Step 1: Identify Key Scalability Attributes:**
  - Define the specific scalability
    - User Growth: Ability to accommodate increased numbers of users or transactions without performance degradation.
    - Data Volume: Handling significant growth in data storage and processing requirements.
    - System Load: Maintaining responsiveness and availability under high workloads and peak usage periods.
    - Geographic Expansion: Supporting expansion into new locations or serving a wider user base.
    - New Features: Facilitating the addition of new features and functionalities without compromising performance.
  - **Step 2: Prioritize Scalability Pain Points:**
    - Review system performance metrics, user feedback, and capacity planning projections to identify potential scalability bottlenecks.
    - Focus on areas that are likely to constrain growth or negatively impact user experience under increased load.
  - **Step 3: Align with Growth Projections:**
    - Ensure the chosen drivers align with anticipated growth in users, data, transactions, or other relevant metrics.
    - Consider future expansion plans and potential spikes in demand.

3. Choose One or More Elements of the System to Refine:

- **Step 1:** Focus on specific areas needing improvement: Identify components of the identity system that require optimization based on scalability.
- **Step 2:** Start with high-impact areas: Prioritize refining components that offer the most significant improvements in scalability.
- **Step 3:** Consider the overall system architecture: Ensure any changes to individual elements align with the overall design and do not introduce any unintended consequences.

4. Choose One or More Design Concepts That Satisfy the Selected Drivers:

- **Step 1:** Research and explore existing solutions: Evaluate existing identity management solutions in healthcare and other industries to identify potential best practices.
- **Step 2:** Consider emerging technologies: Analyze the potential of new technologies, such as blockchain or biometrics, to address specific scalability drivers.
- **Step 3:** Develop new design concepts: When necessary, innovate and develop novel approaches to identity management tailored to healthcare needs.

5. Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces:

- **Step 1:** Implement the selected design concepts: Develop the technical specifications and components based on the chosen design concepts.
- **Step 2:** Allocate responsibilities: Clearly define ownership and accountability for different aspects of the system design and implementation.
- **Step 3:** Define clear interfaces: Establish standardized ways for different components of the identity system to communicate and interact with each other.

6. Sketch Views and Record Design Decisions:

- **Step 1:** Document the design process with diagrams and visual aids: Visually represent the system architecture, components, and relationships through diagrams and visual aids.
- **Step 2:** Record rationale and decisions: Explain the reasoning behind the design choices and the trade-offs made during the design process.
- **Step 3:** Maintain clear documentation: Ensure all documentation is readily accessible and updated throughout the development lifecycle.

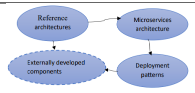
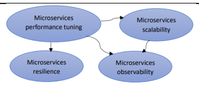
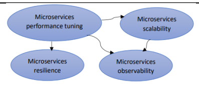
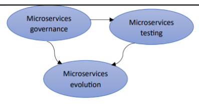
7. Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose:

- **Step 1:** Evaluate the design against the established drivers: Conduct thorough analysis to ensure the design effectively addresses the chosen scalability drivers.
- **Step 2:** Review the achievement of the iteration goal: Assess whether the chosen design effectively addresses the specific goals set for the current iteration.
- **Step 3:** Evaluate the overall design purpose: Ensure the evolving design aligns with the overarching purpose and vision for the identity-based healthcare solution.

Iterate and Continuously Improve:

- We repeat steps 2 to 7 as needed to continuously refine and improve the design based on drivers, feedback, changing requirements, and technological advancements.

Design Roadmap of Identity Based Healthcare Systems for Mature Domains

Iteration	Goal	Design
1	Establish an initial overall system structure	 Reference architectures Microservices architecture Externally developed components Deployment patterns
2	Define the microservices to support primary functionality	 Microservices performance tuning Microservices scalability Microservices resilience Microservices observability
3	Refine previously created microservices to fully address the remaining drivers.	 Microservices performance tuning Microservices scalability Microservices resilience Microservices observability
4	Additional Considerations	 Microservices governance Microservices testing Microservices evolution

Iteration 1

- **Goal:** Establish an initial overall system structure.
- **Design concepts:**
  - **Reference architectures:** Evaluate existing reference architectures for identity management in healthcare, such as the FHIR Identity Framework.
  - **Microservices architecture:** Design the identity-based health solution as a microservices architecture, with each microservice responsible for a specific functionality.
  - **Deployment patterns:** Consider different deployment options for the microservices, such as cloud-based container orchestration platforms or on-premises Kubernetes clusters.
  - **Externally developed components:** Assess the feasibility of using externally developed microservices for specific functionalities, such as authentication, authorization, and user management.

Iteration 2

- **Goal:** Define the microservices to support primary functionality.

- **Design concepts:**
  - **Microservices decomposition:** Identify the core functionalities of the identity-based health solution and decompose them into individual microservices.
  - **Microservices APIs:** Define APIs for each microservice to communicate with each other and with external systems.
  - **Data modeling:** Design the data model for the identity-based health solution, considering the needs of the different microservices.
  - **Microservices security:** Implement robust security measures to protect the microservices and the data they manage.

Iteration 3

- **Goal:** Refine previously created microservices to fully address the remaining drivers.
- **Design concepts:**
  - **Microservices performance tuning:** Optimize the performance of the microservices to improve response times and throughput.
  - **Microservices scalability:** Design the microservices to be scalable to meet the growing needs of the identity-based health solution.
  - **Microservices observability:** Implement monitoring and logging systems to track the performance and health of the microservices.
  - **Microservices resilience:** Design the microservices to be resilient to failures and ensure high availability of the identity-based health solution.

Iteration 4

- **Goal:** Additional Considerations
- **Design concepts:**
  - **Microservices governance:** Develop a governance framework to manage the development, deployment, and operation of the microservices.
  - **Microservices testing:** Implement comprehensive testing strategies to ensure the quality and reliability of the microservices.
  - **Microservices evolution:** Design the microservices to be evolvable, so that new features and functionalities can be easily added or modified in the future.

Conclusion

This document has outlined a comprehensive approach to designing and developing an identitybased healthcare system using Attribute-Driven Design 3.0. The proposed approach emphasizes an iterative development process driven by key security, privacy, scalability, and interoperability drivers. Each iteration emphases on refining specific elements of the system to address these drivers and ensure the solution meets the evolving needs of patients, healthcare providers, and the healthcare ecosystem as a whole. The document also details the selection of a microservices architecture as the foundation for the identity-based healthcare system. This architecture offers plentiful advantages, including scalability, flexibility, and fault isolation, making it well-suited for the complex and dynamic requirements of healthcare. By following the outlined steps and best practices, we can increase the chances of success for our identity-based healthcare solution. This system has the potential to revolutionize healthcare delivery in Bangladesh by improving patient data management, ensuring doctor legitimacy, and empowering patients to make informed choices about their care.