10/1/2023

Design Implementation Proposal for Head in the Cloud IT's Client/Server Project

Jasonrochau\_CS457\_IP5

Jason Rochau

Contents

[Introduction 3](#_Toc147045589)

[Purpose of the System 3](#_Toc147045590)

[General Nature of the System and Software 3](#_Toc147045591)

[Identification of Project Client 4](#_Toc147045592)

[Objectives 5](#_Toc147045593)

[Functional Objectives 5](#_Toc147045594)

[Non-Functional Objectives 5](#_Toc147045595)

[Constraints 7](#_Toc147045596)

[Technical Constraints 7](#_Toc147045597)

[Budgetary Constraints 7](#_Toc147045598)

[Timeline Constraints 8](#_Toc147045599)

[Requirements 9](#_Toc147045600)

[Functional Requirements 9](#_Toc147045601)

[Non-Functional Requirements 9](#_Toc147045602)

[Design Description 11](#_Toc147045603)

[Architecture Overview 11](#_Toc147045604)

[Data Models 12](#_Toc147045605)

[Algorithms and Processes 12](#_Toc147045606)

[Pseudocode: 13](#_Toc147045607)

[Interface Designs 13](#_Toc147045608)

[Diagrams 15](#_Toc147045609)

[Architectural Diagram 15](#_Toc147045610)

[Sequence Diagrams 16](#_Toc147045611)

[Data Migration Sequence Diagram: 16](#_Toc147045612)

[Sequence Diagram (Migration): 16](#_Toc147045613)

[Authentication Sequence Diagram with AuditLog: 17](#_Toc147045614)

[Sequence Diagram (Authentication): 17](#_Toc147045615)

[Sequence 1: User Authentication 18](#_Toc147045616)

[Sequence 2: Accessing Secure Data 19](#_Toc147045617)

[Sequence 3: Logging User Activity 21](#_Toc147045618)

[Risks and Mitigations 23](#_Toc147045619)

[Maintenance and Use Phase Risks 23](#_Toc147045620)

[V&V Plan: 25](#_Toc147045621)

[V&V Plan Overview: 25](#_Toc147045622)

[Conclusion 26](#_Toc147045623)

[Summary of Design Implementation Proposal 26](#_Toc147045624)

[Architecture Overview 26](#_Toc147045625)

[Data Models 26](#_Toc147045626)

[Algorithms and Processes 26](#_Toc147045627)

[Interface Designs 27](#_Toc147045628)

[Diagrams 27](#_Toc147045629)

[Appendix A 28](#_Toc147045630)

[Appendix B 29](#_Toc147045631)

[Appendix C 30](#_Toc147045632)

[References 31](#_Toc147045633)

# Introduction

## Purpose of the System

The purpose of this new client/server system is to migrate our existing on-premises server infrastructure to a more scalable and efficient cloud-based solution. The aim is to transform the way Head in the Cloud IT manages its resources, handles data, and serves its clientele. The new system will capitalize on cloud computing's advantages, including increased scalability, reliability, and cost-efficiency, to ensure a robust, future-proof IT infrastructure.

## General Nature of the System and Software

We're envisioning a multi-tier architecture, ensuring seamless interaction between client applications and cloud-based server systems (Saraswathi, 2020). The system will offer real-time updates, robust backup mechanisms, and a secure operational environment. The software stack will be versatile and layered, including server-side applications for data processing, database management systems for data storage, and client-side user interfaces for interaction and control. Depending on the specific needs of various departments within Head in the Cloud IT, client software may range from lightweight applications for basic tasks to comprehensive dashboard interfaces for more complex operations. Security features are being designed to weave into the system's architecture, thereby ensuring data integrity and confidentiality at all levels.

By transitioning to this cloud-based system, Head in the Cloud IT aims to redefine its operational paradigms and set new benchmarks in operational excellence and client satisfaction (Microsoft 2023, February 28). The use of modern cloud technologies will provide a solid foundation for this transformation.

## Identification of Project Client

The client for this project is our own organization, Head in the Cloud IT. This migration project aims to serve both our internal operational needs and the demands of our external clientele, thereby enhancing overall business efficiency and client service quality (Data Recovery, 2022).

# Objectives

## Functional Objectives

The focal point of the functional objectives is the migration of existing on-premises data and applications to the cloud without any data loss or downtime. The system will also feature strong security measures, like multi-factor authentication (Microsoft. (2023, June 14) and robust data encryption, to safeguard sensitive information. A high level of interactivity will be maintained between client and server through real-time data syncing and efficient request-response mechanisms.

## Non-Functional Objectives

Transitioning to a cloud-based model sets the stage for unprecedented scalability. As Head in the Cloud IT grows, the system will accommodate an expanding user base without compromising performance. Reliability and uptime are also of the essence, with a disaster recovery plan, failover systems, and data backups to ensure seamless operations even under adverse conditions.

Security, always a paramount concern, will be bolstered through cutting-edge encryption methods and real-time intrusion detection systems. In addition, the cloud-based system will undergo regular audits to ensure compliance with data protection laws like GDPR (Violino, B. (2019, May 22), thus safeguarding both the company and its clients.

Performance optimization will be built into the system's DNA, aiming to reduce latency and accelerate data processing speeds. Usability is another cornerstone; the cloud-based client interface will be designed for ease of use, ensuring a smooth transition for all users.

When it comes to maintainability, the cloud-based architecture offers easier upgrades, better bug tracking, and simplified routine maintenance, thereby reducing long-term operational costs. For interoperability, we’ll ensure that our system communicates effortlessly with various types of client hardware and third-party services.

These objectives are designed to ensure that the new client/server cloud-based system aligns perfectly with the strategic goals and operational requirements of Head in the Cloud IT. The system will not only be scalable, secure, and reliable but also optimized for performance, usability, and cost-efficiency, setting the stage for a sustainable future.

# Constraints

## Technical Constraints

Transitioning to a cloud-based architecture often involves rethinking the existing infrastructure, which may not be fully compatible with the new setup. This could require retrofitting or reengineering the old systems, an effort that must be accurately assessed in advance (Cloud Academy Team, 2023).

Older applications were typically not designed with the cloud in mind, leading to integration challenges. The cost and effort to either update or replace these systems need to be clearly defined.

When dealing with data volumes that range from large databases to individual user settings, ensuring that all data transfers correctly without corruption is a massive task. Specialized tools and expert supervision are required for this. This is consistent with the guidelines on Zero Trust Architecture outlined by the Defense Information Systems Agency and the National Security Agency Zero Trust Engineering Team (DISA & NSA, 2022).

## Budgetary Constraints

Although one of the long-term advantages of cloud computing is cost efficiency, the initial setup can be expensive. Costs might include licensing, training, and the purchase of any additional hardware or software needed for the migration.

Customization needs can arise during the migration process that were not initially accounted for. This could include costs for third-party services to handle specific integration challenges (Cloud Academy Team, 2023).

After migration, the operational expenses might differ from the initial estimates. Subscription fees, storage costs, and other recurrent expenditures must be factored into the long-term budget planning.

## Timeline Constraints

Achieving a seamless migration often involves parallel running of old and new systems and a cutover phase, which can be both time-consuming and complex to manage (Gillis, Lelii, & Hefner, n.d.).

The time needed to train the staff and other stakeholders can be significant. These training periods must be strategically placed so they don't interfere with day-to-day operations. Whether due to technical glitches, errors in data migration, or slower-than-expected user adoption, several factors could push back the project timeline (DISA & NSA, 2022).

# Requirements

## Functional Requirements

**Seamless Data Migration:** Migrating data from on-premises to the cloud must happen without any loss or corruption. This will require rigorous pre-testing and the employment of migration tools that validate data integrity post-transfer.

**Real-time Data Syncing:** The cloud system must automatically reflect any changes in data across all platforms and devices. This requires implementing robust data syncing algorithms and technologies to ensure immediate updates IBM. (n.d.).

**User Management:** Administrative personnel should have the capability to manage users with a set of role-based controls. This feature should enable quick onboarding and offboarding, as well as the ability to modify user privileges as required.

**Multi-factor Authentication:** A security protocol involving multiple steps for user authentication, such as passwords, OTPs, and biometrics, must be integrated into the system to deter unauthorized access (Microsoft 2023, June 14).

**Integration with Legacy Systems**: Given that some existing systems will continue to be used, the cloud solution must offer integration capabilities that don't compromise on performance or security.

## Non-Functional Requirements

**Scalability:** As the user base or data volume grows, the system should adapt without causing a slowdown in performance. This will likely involve load-balancing solutions and automated resource allocation.

**Reliability:** A guaranteed uptime of at least 99.9% means the system should have robust failover mechanisms. Any downtime should be minimized, and the system should be capable of rapid recovery.

**Security:** Ensuring the safety of data, it's vital to have end-to-end encryption both when data is in transit and when it's at rest. Moreover, conducting periodic security audits, such as vulnerability assessments and penetration tests, can help keep the system secure from potential threats (Defense Information Systems Agency & National Security Agency Zero Trust Engineering Team, 2022).

**Usability:** An intuitive user interface will facilitate quicker adoption among employees. The design should follow best practices in UX/UI, offering a clean, straightforward path to performing common tasks (Microsoft. 2023, February 28).

**Maintainability:** Updates and maintenance should be able to be performed without taking the system offline for extended periods. This may involve staged rollouts and the ability to roll back updates if they cause issues.

**Interoperability:** The system must be designed to integrate smoothly with different types of client hardware and other third-party services, whether it’s for data exchange, authentication, or other functionalities.

# Design Description

## Architecture Overview

Our cloud-based system is designed to seamlessly migrate existing on-premises data and applications to the cloud while maintaining data integrity and minimizing downtime. The architecture follows a microservices pattern to achieve modularity and scalability (Nandaniya, 2023).

Key components of our system include:

* Data Migration Service: Responsible for orchestrating the migration process. It interfaces with on-premises data sources and cloud storage.
* Real-time Sync Service: Ensures that data changes are propagated in real-time across all platforms and devices, enabling immediate updates for users.
* User Management Module: Provides administrators with role-based controls for managing user accounts, including onboarding, offboarding, and privilege modification.
* Security Layer: Incorporates multi-factor authentication and end-to-end data encryption to fortify the system against unauthorized access and data breaches.
* Legacy Integration Gateway: Allows our cloud solution to integrate seamlessly with existing on-premises systems while maintaining performance and security.

Our design choices align with the project's objectives. The microservices architecture enhances scalability, enabling us to accommodate an expanding user base without performance degradation (Nandaniya, 2023). It also supports our reliability and uptime goals by allowing individual services to be independently deployed and maintained.

## Data Models

Our data models consist of several entities, including "User," "DataRecord," and "AuditLog." The User entity stores user information such as username, email, and role. DataRecord represents individual data entries, while AuditLog records system activities (Anderson, Hall, Hartline, & Hobbs, 2010).

## Algorithms and Processes

Data Migration Process:

1. Users select the source of data from on-premises systems or external storage.
2. Data from the selected source is pre-tested for compatibility with the cloud environment. Any issues are flagged for resolution (Anderson, Hall, Hartline, & Hobbs, 2010).
3. The migration service orchestrates the transfer of data from the source to the cloud, using migration tools that validate data integrity post-transfer (Anderson, Hall, Hartline, & Hobbs, 2010).
4. After migration, the system validates the integrity of migrated data to ensure no loss or corruption occurred during the process (Anderson, Hall, Hartline, & Hobbs, 2010)..

### Pseudocode:

A screenshot of a computer program

Description automatically generated

## Interface Designs

Our user interface is designed with an efficient layout, catering to the preferences of our technology-savvy users. It mirrors the familiarity of a terminal or PowerShell environment, prioritizing functionality and ease of use. Please see the Appendices for the screenshots of the mentioned layouts.

**Command Dashboard:** Upon login, users are presented with a command dashboard, reminiscent of a terminal interface. This dashboard provides a concise summary of system activities and data migration progress, displayed in a structured text format.

**Data Migration Wizard:** Our data migration wizard is implemented as a series of command-line prompts. Users are guided through the migration process step by step, with each command providing clear instructions and progress indicators in a terminal-style interface.

By adopting this terminal-like interface, we aim to cater to the technical expertise of our user base while maintaining a user-friendly and efficient user experience.

# Diagrams

## Architectural Diagram

Purpose: The architectural diagram illustrates the high-level structure of our new client/server system. It shows how different components interact to achieve our migration objectives (Nandaniya, 2023). It's a visual representation of the system's architecture.

Components:

User Interface: This is the part of the system that users interact with directly.

Application: The application layer handles the processing of data and logic.

Database: This is where data is stored.

Interactions: The arrows or lines in the diagram represent interactions between these components. For example, user requests flow from the User Interface to the Application, which then interacts with the Database.

+----------------------+

| User Interface |

+---------------------+

|

+---------------------+

| Application |

+---------------------+

|

+--------------------+

| Database |

+--------------------+

## Sequence Diagrams

Sequence diagrams provide a time-ordered view of interactions within the system. It shows the order of messages and how control flows between different objects or components (Visual Paradigm, 2023, September).

**Participants:**

* User: Represents the end-user interacting with the system.
* Application: Represents the software application.
* Database: Represents the database system.

**Interactions: In the sequence diagram:**

1. The User initiates a request (e.g., data retrieval) to the Application.
2. The Application processes the request and queries the Database.
3. The Database responds to the Application.
4. Finally, the Application presents the result to the User.

### Data Migration Sequence Diagram:

1. User requests the initiation of data migration.
2. The Application processes the request and initiates the Data Migration Service.
3. The Data Migration Service communicates with the source (on-premises system) and fetches the DataRecords.
4. As the Data Migration Service processes these records, any system activities and actions taken on the DataRecords are logged into the AuditLog.
5. Post migration, the Data Migration Service confirms the successful transfer of DataRecords and updates the AuditLog accordingly.

### Sequence Diagram (Migration):

1. User --> Application: Initiate Data Migration
2. Application --> Data Migration Service: Start Migration
3. Data Migration Service --> On-premises System: Fetch DataRecords
4. Data Migration Service --> AuditLog: Log Migration Activity
5. On-premises System --> Data Migration Service: Send DataRecords
6. Data Migration Service --> Cloud Database: Store DataRecords
7. Cloud Database --> Data Migration Service: Confirm Successful Storage
8. Data Migration Service --> AuditLog: Log Successful Migration
9. Data Migration Service --> User: Confirm Migration Completion

### Authentication Sequence Diagram with AuditLog:

1. The User requests authentication from the Authentication Server.
2. The Authentication Server requests credentials from the User.
3. The User provides the necessary credentials to the Authentication Server.
4. The Authentication Server verifies these credentials and returns the authentication result.
5. All authentication activities, including successful and failed attempts, are recorded in the AuditLog.

### Sequence Diagram (Authentication):

1. User --> Authentication Server: Request Authentication
2. Authentication Server --> User: Request Credentials
3. User --> Authentication Server: Submit Credentials (username, password)
4. Authentication Server --> AuditLog: Log Authentication Attempt
5. Authentication Server --> Authentication Server: Verify Credentials
6. Authentication Server --> User: Authentication Result (authenticated, token)
7. Authentication Server --> AuditLog: Log Authentication Outcome

# Sequence 1: User Authentication

A diagram of a process

Description automatically generated

Description of Sequence 1:

* The ‘User’ sends a ‘requestAuthentication’ message to the Authentication Server.
* In response, the Authentication Server sends back a ‘requestCredentials’ message to the User, asking for the required authentication credentials.
* The User then sends the credentials, encapsulated in the ‘submitCredentials’ (username, password) message, to the Authentication Server.
* The Authentication Server internally processes these credentials using the ‘verifyCredentials’ action.
* Based on the verification, the Authentication Server sends an ‘authenticationResult’ (authenticated, token) message back to the User. This message will either confirm successful authentication (and possibly provide an authentication token) or inform the User of a failed authentication attempt.

# Sequence 2: Accessing Secure Data

A diagram of a computer

Description automatically generated

Description of Sequence 2:

* The User sends a ‘requestSecureData’ message with a token to the Data Server.
* The Data Server internally checks the validity of the token with the ‘verifyTokenValidity’ process.
* 2a. If the token is valid, the Data Server sends a ‘tokenValid’ message to the User (Auth0, 2023, September).
* 2b. If the token is invalid, the Data Server sends a ‘tokenInvalid’ message to the User and proceeds to send a ‘logInvalidTokenEvent’ (with the User's details) to the Log Server.
* The Log Server receives the ‘logInvalidTokenEvent’ and stores the information.
* If the token was deemed valid in step 2a, the User then sends a ‘specifyDataRequest’ (dataID) message to the Data Server (Auth0, 2023, September).
* The Data Server processes the data request and sends back the data to the User using the ‘provideDataResponse’ (data) message.

# Sequence 3: Logging User Activity

A diagram of a computer

Description automatically generated

Description of Sequence 3:

* User -> Logging Server: ‘initiateLogAction’ (actionType)
* Depending on the context, ‘actionType’ could be authenticationResult or ‘tokenValidation’.
* Logging Server: ‘determineLogCategory’ (actionType)
* This is an internal process to determine where or how to log the event based on the action type.
* Logging Server -> User: ‘requestAdditionalInfo’
* This could be an optional step, where the Logging Server might require more details about the action for thorough logging.
* User -> Logging Server: ‘provideAdditionalInfo’ (details)
* Here, the user provides additional information (like a timestamp, IP address, or other relevant details).
* Logging Server: ‘storeLogDetails’ (actionType, details)
* An internal action where the Logging Server ensures the log details are stored correctly.
* If integrating with the Authentication Server or Data Server:
* Authentication Server/Data Server -> Logging Server: ‘forwardLogDetails’ (details)
* The server sends details of the interaction to be logged. This step captures logging interactions originating from system components rather than direct user actions.

Logging Server: ‘processForwardedDetails’ (details)

# Risks and Mitigations

In this context, we present a comprehensive overview of risks and their corresponding mitigation strategies that are vital for the seamless functioning of the system during its maintenance and use phases (Nishadha, 2023). These risks span a spectrum of possibilities, from data corruption during routine maintenance to unforeseen hardware failures and user resistance to change. Each risk is accompanied by a strategic mitigation plan designed to minimize potential disruptions and ensure the software's long-term success (Nishadha, 2023).

By acknowledging these risks and proactively addressing them, organizations can fortify their software systems, enhance their reliability, and uphold a commitment to delivering uninterrupted services to their users. In essence, the mitigation strategies presented here serve as a shield against the uncertainties that may arise, allowing organizations to navigate the complexities of software maintenance and use with confidence and resilience Nishadha, 2023).

## Maintenance and Use Phase Risks

| **Risk** | **Mitigation** |
| --- | --- |
| Data Corruption During Routine Maintenance | Implement thorough backup and recovery mechanisms. Regularly test data backups and ensure that they are easily accessible for restoration in case of data corruption, "implement thorough backup and recovery mechanisms" (RiskOptics, 2021). |
| Increased Operational Costs | Continuously monitor operational expenses and conduct regular cost-benefit analyses. Implement cost optimization strategies, such as resource scaling and efficient resource allocation. |
| Security Vulnerabilities | Conduct regular security audits and penetration testing. Keep software and security libraries up to date with patches and updates. Implement intrusion detection systems and real-time monitoring. |
| User Resistance to Change | Provide comprehensive user training and support during the transition to the new system. Gather user feedback and make iterative improvements based on their needs and concerns, "provide comprehensive user training and support during the transition to the new system" (RiskOptics, 2021). |
| Technical Obsolescence | Plan for technology upgrades and system modernization in advance. Keep abreast of emerging technologies and industry best practices to stay relevant. |
| Lack of Documentation | Create and uphold an extensive system documentation library encompassing user manuals, technical guides, and system architecture documents. Guarantee that knowledge is not only documented but also easily accessible to the entire team. To mitigate the risk of a lack of documentation, it is crucial to "create and uphold an extensive system documentation library encompassing user manuals, technical guides, and system architecture documents" (RiskOptics, 2021). |
| Unforeseen Hardware Failures | Implement redundancy and failover mechanisms for critical hardware components. Regularly test and replace hardware as needed to minimize the risk of unexpected failures. |
| Compliance Violations | Stay updated on relevant regulations and compliance requirements. Implement audit trails and reporting mechanisms to ensure continuous compliance monitoring. |
| Inadequate User Support | "Establish a robust user support system, including helpdesk services and user forums" (RiskOptics, 2021). Ensure that users have access to timely and effective technical support. |

# Verification and Validation (V&V) Plan:

## V&V Plan Overview:

* **Organization:**

The V&V team is a group of testers but a combination of testers and developers. This mix ensures that the team can understand both the intent behind the design and the potential pitfalls. Experienced members of both professions bring a depth of knowledge, ensuring comprehensive coverage during testing phases (SMC Systems Engineering Handbook, 2023).

* **Schedule:**

Running V&V activities concurrently with the design phase offers several advantages. This overlap ensures immediate feedback on design choices (pp\_pankaj, 2023), enabling swift course corrections if needed. It also aids in identifying potential issues early in the development lifecycle, reducing costs and delays that might arise if problems are detected later.

* **Resources:**

Our V&V plan is backed by substantial resources:

* + Dedicated test environments ensure that testing is conducted in isolated conditions, replicating real-world scenarios without affecting ongoing development (pp\_pankaj, 2023).
  + We've invested in top-tier testing tools to automate repetitive tasks and ensure thorough coverage.
  + Skilled personnel are the backbone of our V&V activities. Their expertise ensures that our software meets the highest standards.
* **Responsibilities**:

By dividing responsibilities, we make sure every level of our software is scrutinized:

* + Developers take charge of unit testing. Their intimate knowledge of the codebase ensures that individual modules function correctly (SMC Systems Engineering Handbook, 2023).
  + Testers handle system testing, focusing on the software, its interactions with external systems, and its behavior under various conditions.
* **Tools:**

We employ a suite of modern tools to aid our V&V activities:

* + Testing frameworks cater to both frontend and backend components, ensuring consistency (SMC Systems Engineering Handbook, 2023).
  + Automated testing tools help in regression testing and load testing, ensuring the software remains robust as it evolves.
  + Bug tracking systems like Bugzilla (SMC Systems Engineering Handbook, 2023) help in logging, tracking, and managing defects efficiently.
* **Techniques:**

Our techniques cover a broad spectrum of testing types:

* + White-box testing ensures the internal structures of the application are sound.
  + Black-box testing verifies the software's external functionalities.
  + Performance testing ensures the software remains responsive under load (SMC Systems Engineering Handbook, 2023).
  + Security testing is pivotal, ensuring that the software is resilient against potential threats and vulnerabilities.
* **Methodologies:**

Adopting an Agile approach allows us to remain flexible and responsive to changes development (pp\_pankaj, 2023). With continuous integration, every code change triggers a series of tests, ensuring that new additions don't introduce regressions (SMC Systems Engineering Handbook, 2023). This continuous testing approach ensures that Head In the Cloud’s software remains reliable throughout the development lifecycle.

# Conclusion

## Summary of Design Implementation Proposal

### Architecture Overview

The system architecture leverages a microservices design pattern. This approach breaks down the application into smaller, independently deployable services. Each service is responsible for a distinct functionality and operates in isolation. This design ensures modularity, enabling individual components to scale or be updated independently, without affecting the entire system. The microservices pattern, combined with container orchestration tools like Kubernetes (Ozkaya, 2021), provides Head In the Cloud IT, the advantage of scalability, fault tolerance, and a streamlined deployment process.

### Data Models

Our system encapsulates several primary data entities:

* **User:** This entity stores information about the end-users, including credentials, user preferences, and activity history.
* **DataRecord:** Central to our application, the DataRecord encompasses the actual data units being transferred or manipulated. It may include fields like dataID, content, timestamp, and source.
* **AuditLog:** Serving as an immutable ledger, the AuditLog captures all critical system interactions, including data migrations, user authentication events, and potential system errors. This ensures traceability and accountability within the application.

### Algorithms and Processes

One of the pivotal processes in our system is the data migration process.

* Fetching records from a source, often an on-premises system.
* Validating and possibly transforming the data.
* Logging the migration event to the AuditLog.
* Storing the data in its new destination, typically a cloud-based database.

**Pseudocode**

FUNCTION migrateData(source, destination):

records = source.fetchRecords()

FOR record IN records:

isValid, transformedRecord = validateAndTransform(record)

IF isValid:

destination.store(transformedRecord)

AuditLog.logEvent("Data Migrated", record.dataID)

### Interface Designs

The system's user interface primarily adopts a terminal-like layout. Recognizing the efficiency of command-line interfaces, especially for users familiar with technical systems, we've designed an interface that reduces GUI complexities without sacrificing functionality. Users can navigate using commands through a dashboard, and when initiating processes like data migration, a step-by-step wizard assists them. Initial feedback from user testing has shown a preference for this streamlined design, allowing users to perform tasks rapidly and accurately.

### Diagrams

Our documentation is supplemented by architectural and sequence diagrams. Sequence diagrams, on the other hand, provide a temporal representation of interactions. They showcase the order in which operations occur, the participants involved, and the messages exchanged. For instance, the "User Authentication" sequence diagram elaborates on the steps involved when a user attempts to authenticate, highlighting the interactions between the user, the application, and the authentication server.

# Appendix A

A screenshot of a computer

Description automatically generated

# Appendix B

A screenshot of a computer

Description automatically generated

# Appendix C

A screenshot of a computer program

Description automatically generated

# References

Microsoft. (2023, June 14). Passwordless authentication options for Azure Active Directory. Retrieved from <https://learn.microsoft.com/en-us/azure/active-directory/authentication/concept-authentication-passwordless#microsoft-authenticator>

Violino, B. (2019, May 22). How to conduct a proper GDPR audit: 4 key steps. CSO Online. Retrieved from <https://www.csoonline.com/article/573755/how-to-conduct-a-proper-gdpr-audit-4-key-steps.html>

Saraswathi, R. (2020, January 6). Four Architecture Choices for Application Development in the Digital Age. IBM Blog. <https://www.ibm.com/blog/four-architecture-choices-for-application-development/>

Microsoft. (2023, February 28). Cloud center of excellence (CCoE) functions. Learn Microsoft. <https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/organize/cloud-center-of-excellence>

Data Recovery. (2022, June 6). Defining Key Stakeholders in a Data Migration Project. <https://datarecovery.com/rd/defining-key-stakeholders-in-data-migration/>

Cloud Academy Team. (2023, July 21). What is Cloud Migration? Strategy, Processes, Benefits & Risks. Cloud Academy. <https://cloudacademy.com/blog/cloud-migration-benefits-risks/>

Defense Information Systems Agency & National Security Agency Zero Trust Engineering Team. (2022, July). Zero Trust Reference Architecture Version 2.0. Department of Defense. <https://dodcio.defense.gov/Portals/0/Documents/Library/(U)ZT_RA_v2.0(U)_Sep22.pdf>

Gillis, A. S., Lelii, S., & Hefner, K. (n.d.). Data migration. TechTarget. <https://www.techtarget.com/searchstorage/definition/data-migration>

Visual Paradigm. (2023, September). What is Sequence Diagram? Retrieved from <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>

IBM. (n.d.). Cloud migration planning with IBM Turbonomic. Retrieved [Insert date you accessed this webpage], from <https://www.ibm.com/products/turbonomic/cloud-migration-planning?utm_content=SRCWW&p1=Search&p4=43700076774885473&p5=p&gclid=Cj0KCQjw0vWnBhC6ARIsAJpJM6fYbBacQEIvPpalgmB_JNTHj__zjRwclyo1bcyJsILtWWXcHxkmgpkaAo1NEALw_wcB&gclsrc=aw.ds>

Nandaniya, H. (2023, August 9). Software Architecture Patterns: Driving Scalability and Performance. Maruti Tech. Retrieved from <https://marutitech.com/software-architecture-patterns/>

Anderson, E., Hall, J., Hartline, J. D., & Hobbs, M. (February 2010). Algorithms for Data Migration. Algorithmica, 57(2), 349-380. DOI: 10.1007/s00453-008-9214-y.

pp\_pankaj. (2023, September 20). Verification and Validation in Software Engineering. GeeksforGeeks. <https://www.geeksforgeeks.org/software-engineering-verification-and-validation/>

SMC Systems Engineering Handbook – Chapter 7 Modeling and Simulation Verification and Validation Challenges. (2023, July). Retrieved from AcqLinks website: <https://acqnotes.com/acqnote/careerfields/verification-validation>

RiskOptics. (2021, August 12). Risk Mitigation in Software Engineering. <https://reciprocity.com/risk-mitigation-in-software-engineering/>