PROJECT NEURON

HIGH-LEVEL ARCHITECTURE

V0.2

TABLE OF CONTENTS

INTRODUCTION

This document provides a high-level overview of the proposed solution architecture for Willis Towers Watson's **Neuron** platform, addressing business needs and automating insurance risk management processes. The Neuron platform's design principles are a flexible, scalable, and robust **Software as a Service (SaaS)** solution that aims to streamline auto-follow practices in insurance risk management while introducing a platform for efficient collaboration between insurers and brokers.

The Neuron platform is envisioned as a predominantly API-driven, headless service platform, enabling multiple parties to manage auto-follow risk assessments efficiently and quotes through an open set of APIs. In a headless architecture, the Platform's core functionalities are decoupled from the frontend applications, allowing various parties to access and interact using their custom frontend interfaces. This approach simplifies integration with diverse systems and ensures seamless access to the Platform across different devices and technologies.

In addition to the API-driven model, Neuron will feature a user interface (UI) designed explicitly for brokers and support users. This UI will provide a more user-friendly and interactive experience for managing and monitoring auto-follow risks, quotes, and collaborations between insurers and brokers.

INTENDED AUDIENCE

The intended audience for this high-level architecture documentation includes:

- Project managers need to understand the overall design and architecture of the system to plan and manage the project effectively, and
- Business stakeholders such as product owners or business analysts need to understand the system's overall design and architecture to ensure that it aligns with the business requirements, and
- Technical stakeholders such as developers, architects, and testers need to understand the overall design and architecture of the system to implement, test, and maintain it, and
- Operations and maintenance staff who will benefit from understanding the system's components and their interactions to understand the system's maintenance requirements, and
- Regulators may need to review the documentation to ensure that the system meets the relevant standards and regulations, and
- Future developers may need to maintain or enhance the system, as it provides a clear picture of its overall design and architecture.

SCOPE

The scope of this high-level architecture documentation is to provide an overview of the overall design and architecture of Project Neuron. This documentation will cover the system's key components, their interactions, and how they support its functionality. The system will be designed to meet the requirements listed under Business Requirements.

IN SCOPE

1. Overview: A summary of the system's purpose and goals, as well as its main features and functions, and
2. System context: A description of the system's environment and the external systems or stakeholders with which it will interact, and

3. Key components: A high-level description of the system's key components, including hardware, software, and data components, and
4. Interaction diagrams: Diagrams that show how the key components interact and support the system's overall functionality, and
5. Data flow diagrams: Diagrams that show how data flows through the system, from input to output, and
6. Security: A high-level overview of the security measures that will be in place to protect the system and its data, and
7. Performance: A high-level overview of the system's expected performance characteristics, such as response times, throughput, and capacity, and
8. Scalability: A high-level overview of the system's scalability, including how it can handle increased workloads and the potential for future expansion, and
9. Non-Functional Requirements: A high-level overview of the system's non-functional requirements, such as high availability, maintainability, and monitoring, and
10. Standards & Compliance: Description of the standards and compliance that the system must adhere to, and
11. Glossary: A list of terms and definitions used throughout the documentation to ensure consistency and clarity.

NOT IN SCOPE

1. Detailed design specifications: This document does not include detailed design specifications for each component or subsystem, and
2. Implementation details: The documentation does not include specific implementation details, such as programming languages, libraries, or frameworks that will be used to implement the system, and
3. Test plans: The documentation does not include detailed test plans or cases that will be used to test the system, and
4. Deployment details: The documentation does not include specific deployment details, such as the particular infrastructure or hosting environment used to deploy the system, and
5. Low-level diagrams: The documentation does not include low-level diagrams that show the internal workings of each component or subsystem, and
6. Code snippets: The documentation does not include code snippets or sample code to demonstrate how the system will be implemented, and
7. User Manuals or Guides: The documentation does not include end-user instructions for using the system, and
8. Project management details: The documentation does not include project management details such as project plans, timelines, or resource allocation.

Version Information

Version	*Purpose/Change*	*Author(s)*	*Date*
DRAFT 0.1	Initial draft	Aldrine Einsteen	18/01/2023
V0.1	Basic version	Aldrine Einsteen	17/03/2023
V0.2	Changes to Monitoring and Authentication	Aldrine Einsteen	05/04/2023

Version Details

Version	*Details*
Draft 0.1	The initial draft of the document, including essential building components.
V0.1	A basic version of the High-Level Architecture Document with information on all building blocks and target architecture. The key changes are the section 1. System Context 2. Container 3. Component 4. Security
V0.2	The latest updates on the decision to use OIDC and Non-Functional requirements are documented.
V0.3	The vestigial data and documentation are removed.

Stake Holders

Role	*Name*	*RACI*
Security Architect	1. Johri, Vivek (Reigate) vivek.johri@wtwco.com	Responsible
Head of Security	1. Stanbury, Kate (Reigate) Kate.Stanbury@wtwco.com	Accountable
Business and Technical Ops	1. Moore, Leigh (London) leigh.moore@wtwco.com	Responsible
Peer Reviewers	1. Johri, Vivek (Reigate) vivek.johri@wtwco.com 2. Vivek Kumar (External) vivek_kumar@epam.com	Consulted
Delivery Manager	1. Shevchenko, Kate (Reigate) kate.shevchenko@wtwco.com	Informed
	2. Cordingley, Justin (Reigate) justin.cordingley@wtwco.com	Informed
Product Manager	1. Soh, Eileen (Reigate) eileen.soh@wtwco.com	Informed
Head of Design and Architecture	1. Morris-Tarry, Alex (Reigate) alex.morris-tarry@wtwco.com	Accountable
Head of Product Design	1. Fairless, Jonty (London) jonty.fairless@wtwco.com	Responsible
User Interface Solution Architect	1. Mike Devenish mike_devenish@epam.com	Consulted
Enterprise Architect	1. Wilson, Ken (Philadelphia) ken.wilson@wtwco.com	Informed
Global Privacy Team	1. Andy Brown andy.brown@wtwco.com	Consulted

EXECUTIVE SUMMARY

"Neuron" aims to revolutionise the insurance risk management industry by streamlining auto-follow practices and providing a platform for seamless collaboration between insurers and brokers. This high-level architecture document overviews the proposed solution, outlining its key components, interactions, and design principles.

"Neuron" is designed as a flexible, scalable, and robust Software as a Service (SaaS) solution, leveraging a predominantly API-driven, headless service platform. This approach enables efficient management of auto-follow risk assessments and quotes while offering seamless integration with diverse systems and technologies. To enhance user experience, Neuron also features a dedicated user interface for brokers and support users.

Commented [EA(1]: The user interface for support and administration (especially for onboarding)

Commented [E(2R1]: @Soh, Eileen (Reigate) If Support UI is not in scope - we can remove and clean up all the reference to Support.

Neuron's architecture focuses on meeting functional and non-functional requirements, including scalability, high availability, maintainability, security, and monitoring. The system is designed to comply with industry standards and regulatory requirements, ensuring a secure and reliable solution for all stakeholders.

From an initial viewpoint of being a headless approach, we have evolved to have an incorporated user-friendly interface for brokers and administration. The paradigm shift primarily aims to break away from all the drag-factors of existing process. Neuron aims to provide a future-proof solution capable of supporting the dynamic landscape of the insurance risk management market. This document serves as a blueprint for developing and implementing the Neuron platform, providing stakeholders with a comprehensive understanding of the system's design, goals, and functionalities.

BUSINESS OVERVIEW

Neuron aims to revolutionise the commercial lines insurance industry by providing a cutting-edge technology platform that enables brokers and insurers to connect digitally more efficiently and at scale. This Platform streamlines communication automates processes, and reduces operational costs for all parties involved.

DRIVERS

1. Inefficient communication: Current communication methods rely heavily on physical documents and fragmented digital formats across multiple email chains.
2. Market demand: Neuron's ability to meet the growing demand for efficient and seamless digital solutions in the insurance industry.
3. Lower operational costs: Implementing cost-saving measures such as process automation and outsourcing to reduce overall operating costs for brokers and insurers.
4. Increased efficiency: Improving operations' efficiency by reducing costs, enhancing productivity and higher profit margins for service providers.
5. Increased competitiveness: Cost reduction enables service providers to offer their services at lower prices, increasing competitiveness in the marketplace and attracting more customers.
6. Improved reachability: Expanding the availability of the insurer pool for brokers by minimising the cost of discovering and establishing relationships through standardisation, enabling access to a broader range of insurers.

GOALS

Functional goals

1. Increase efficiency: The system aims to automate processes, reduce errors and manual labour, and streamline workflows to increase organisational efficiency across brokers and insurers, especially in the auto-follow market.
2. Optimise revenue: Neuron seeks to enhance revenue generation for Willis Towers Watson by streamlining operations and improving overall productivity.
3. Increase competitiveness: The system can provide a competitive advantage by leveraging advanced technologies, data analytics, and other business capabilities.

Technology goals

4. Scalability: The system can adapt to new business requirements and grow with the organisation over time.
5. Reusability: Reusing the back-end functionality across different frontend interfaces.
6. Flexibility: The ability to quickly make changes to the front end or back-end of the application without affecting the other.
7. Speed: Faster loading times for the front end by reducing the data needed.
8. Security: Keeping sensitive data and business logic separate from the front end, reducing the risk of security breaches.
9. Cost-effectiveness: Reducing the cost of development and maintenance by separating the frontend and back end.

Commented [EA(3]: @Soh, Eileen (Reigate) My suggestions are added, but I lack the business knowledge to quantify them.

Feel free to add more – and modify even down the technology goals.

Commented [EA(4]: @Soh, Eileen (Reigate) I have added goals from technology, can you look at the business perspective of the product?

10. Separation of concerns: Each application part can be developed, tested, deployed and scaled independently.

Constraints

1. Regulatory and business constraints: The Neuron platform must adhere to any specific regulatory requirements or business constraints that may be applicable.
2. Dependency on MuleSoft: The Business Tier, including the central logical block, is provisioned as a service from MuleSoft, creating a dependency on their Platform and services.
3. UI deployment: The Neuron user interface is deployed on Azure using Azure DevOps and Azure containers. Due to existing investments and expertise in Azure, this deployment approach is a constraint and the need to ensure consistency in deployment practices across the Platform.
4. Approved CI/CD: Azure DevOps is the approved toolset for CI/CD pipeline management, including Azure (UI) and AWS (MuleSoft) deployment.

Business Requirements

The following section outlines the essential business requirements of Neuron platform must fulfil to successfully streamline auto-follow practices in insurance risk management and foster efficient collaboration between brokers and insurers.

1. User Management: The Platform must provide a robust user management system, allowing for the creation, modification, and deletion of user accounts, along with role-based access control to ensure that users have appropriate permissions to access platform features and data. The best would be the use case where the user-management can be self-serviced by the global admins of the respective organisations.
2. Auto-Follow Risk Assessment: Neuron must provide a seamless process for brokers and insurers to automatically follow risk assessments, enabling them to efficiently evaluate and manage risks associated with insurance policies.
3. Quote Generation and Management: The Platform must facilitate the generation, sharing, and management of insurance quotes between brokers and insurers. It should enable users to easily compare quotes, accept or reject proposals, and track the status of ongoing negotiations.
4. Reporting and Analytics: Neuron should provide comprehensive reporting and analytics features, enabling users to generate insights on risk management, quote performance, and other critical business metrics to support data-driven decision-making.
5. Integration Capabilities: The Platform must support integration with existing systems and tools used by brokers and insurers, ensuring seamless data exchange and minimising the need for manual data entry.
6. Compliance and Regulatory Requirements: Neuron must adhere to all requirements, including data privacy and security regulations, to protect user data and ensure the Platform's lawful operation.
7. Customisation and Branding: The Platform should allow for customisation and branding opportunities, enabling brokers and insurers to tailor the user interface and experience to align with their needs and preferences.
8. Training and Support: Neuron must provide adequate training and support resources, such as user guides, video tutorials, and helpdesk support, to assist users in navigating and utilising the Platform effectively.

Commented [JF5]: TBC @Einsteen, Aldrine (Reigate) - if brokerages are using the UI, they're effectively agreeing to *not* integrate the service closely into their toolset. I'm not sure why we'd then offer to brand it for them - so a TBC requirement

Commented [E(6R5]: @Fairless, Jonty (London) This is mainly to customise the portal to suit Broker vs Insurer (if it ever come true).

So, right now we are pushing the risk to RadarLive SaaS. But there is also a parallel stream of thought - where the Insurer or Insurer system will come and get the risks/quotes for action.

By addressing these business requirements, the Neuron platform will successfully support the digital transformation of auto-follow practices in insurance risk management, offering a comprehensive solution that fosters efficiency, collaboration, and growth for brokers and insurers.

Data Privacy Requirement

Data privacy and the protection of personally identifiable information (PII) are critical concerns for the Neuron platform, given its role in facilitating collaboration between brokers and insurers in the insurance risk management industry. To address these concerns and ensure compliance with relevant data protection regulations, such as the General Data Protection Regulation (GDPR) and other regional privacy laws, the following measures will be implemented:

1. Data Collection and Storage: Neuron will collect, store, and process only the minimum necessary PII data for its intended purpose. Data storage will be secure and encrypted, with access restricted to authorised personnel.
2. Data Retention: The Platform will adhere to strict data retention policies, ensuring that PII data is only stored for the purposes it was collected. Regular audits will be performed to identify and delete any data that is no longer needed.
3. Consent Management: Neuron will incorporate a consent management system, allowing users to provide, withdraw, or modify their consent for data collection, processing, and sharing, following applicable data protection regulations.
4. Access Control and Authentication: The Platform will implement robust access control mechanisms, such as role-based access control and multi-factor authentication, to restrict access to PII data and ensure that only authorised users can access the information.
5. Data Encryption: All PII data transmitted or stored within the Neuron platform will be encrypted, both in transit and at rest, using industry-standard encryption algorithms and protocols.
6. Data Sharing and Third-Party Integrations: The Platform will ensure that PII data is only shared with authorised third parties and only when necessary for the intended purpose. Any third-party integrations will be vetted to comply with data protection standards and privacy regulations.
7. Data Breach Detection and Response: Neuron will implement a comprehensive data breach detection and response plan, including monitoring for potential threats, regular security audits, and incident response procedures to mitigate the impact of a data breach and comply with regulatory notification requirements.
8. Privacy by Design: The Platform will follow the principles of privacy by design, ensuring that data privacy considerations are embedded in every stage of development, from initial design to implementation and maintenance.
9. Staff Training and Awareness: Neuron will provide regular training and awareness programs to ensure that all employees and contractors involved in the Platform's development, operation, and maintenance understand the importance of data privacy and the best practices for protecting PII information.
10. Data Subject Rights: The Platform will facilitate users' ability to exercise their data subject rights, such as the right to access, rectify, erase, or object to the processing of their PII data, following applicable privacy laws.

11. By implementing these measures, Neuron will ensure the highest level of data privacy and protection for its users, safeguarding PII information and maintaining compliance with relevant data protection regulations.

Note: After detailed analysis by the data architect and the appropriate team; For the Release 1 (SOMA), there are no PII information within the application. As for the user information we collect only the work-email address and profile information (which are not PII).

TECHNICAL ARCHITECTURE

OVERVIEW

The technical architecture of Neuron is designed to provide a robust, flexible, and efficient platform that facilitates seamless collaboration between brokers and insurers in the insurance risk management industry. By employing the C4 model for visualising software architecture, the Platform's design focuses on key architecture characteristics that promote modularity, scalability, and adaptability, ensuring the system can evolve with changing business requirements and technological advancements.

In addition to the core principles, the technical architecture of Neuron leverages the capabilities of MuleSoft, a leading integration platform, to optimise performance, streamline development processes, and enhance maintainability. MuleSoft's Anypoint Platform simplifies the design, deployment, and management of APIs and integrations, enabling Neuron to connect seamlessly with various systems and data sources across the insurance risk management ecosystem. The Platform also offers built-in security features, ensuring compliance with relevant data protection regulations.

The technical architecture of Neuron aims to strike a balance between innovation and reliability, providing a robust and future-proof solution that effectively addresses the needs of brokers, insurers, and other stakeholders in the insurance risk management space. The Platform's focus on flexibility, interoperability, and scalability, powered by MuleSoft's integration capabilities, ensures it can seamlessly integrate with existing systems, adapt to new business requirements, and grow with the organisation over time delivering long-term value and a competitive edge in the marketplace.

PRINCIPLES

1. API-first design: Design systems and components with APIs as the primary focus, ensuring ease of access and integration for other systems and components. This approach enables the flexibility to swap services when needed and prevents vendor lock-in.
2. Modularity: Design components and systems modularly, allowing for easy replacement or upgrades without affecting the rest of the system. This approach prevents vendor lock-in by enabling the interchangeability of components.
3. Vendor-agnostic design: Develop the system with components and infrastructure that distribute risk across multiple vendors—Utilise cloud-agnostic vendors to spread risk across various providers and maintain flexibility.
4. Microservices architecture: Design systems and components around small, independent services that can be easily replaced or upgraded without affecting the rest of the system. This approach helps to prevent vendor lock-in by allowing for component interchangeability.
5. Abstraction: Design systems and components with easily understood and accessible interfaces, regardless of the underlying technology. This principle promotes flexibility and prevents vendor lock-in by allowing for component interchangeability.
6. Loose Coupling: Design systems and components to work independently without direct dependencies. This approach helps to prevent vendor lock-in by allowing for component interchangeability.
7. Scalability: Design systems and components capable of handling increasing workloads without significant changes to the architecture. This approach helps to prevent vendor lock-in by allowing for component interchangeability.

Commented [EA(7]: @Wilson, Ken (Philadelphia) – I have put my thoughts into my architectural principles. These are conventional wisdom (as 2023 rolled in – there is a significant emphasis on vendor lock-ins).

8. Redundancy: Design systems and components with multiple copies of data and functionality to ensure continuous operation during failure. This approach helps to prevent vendor lock-in by allowing for component interchangeability.
9. Interoperability: Design systems and components to work seamlessly, regardless of vendor or technology. This approach helps to prevent vendor lock-in by allowing for component interchangeability.
10. Standards-based: Design systems and components that adhere to industry standards, ensuring they can be easily understood and accessed, regardless of the underlying technology. This approach helps to prevent vendor lock-in by allowing for component interchangeability.

By adhering to these principles, the technical architecture of Neuron will ensure a flexible, scalable, and robust platform that meets the needs of its users and adapts to the ever-changing landscape of the insurance industry.

System Context

The System Context section of the C4 visualisation provides an overview of the Neuron's environment, external systems, and stakeholders with which it interacts. This high-level view of the system helps to establish boundaries and clarify the responsibilities of Neuron within the insurance risk management ecosystem. Neuron is designed as a multi-tenant platform, serving multiple brokers and insurers on a single instance.

EXTERNAL SYSTEMS

1. Broker Systems: These systems represent the various software applications and tools insurance brokers use to manage clients, policies, and transactions. Neuron integrates with broker systems through APIs, enabling brokers to access the marketplace and auto-follow features.
2. Insurer Systems: These systems represent the automated software applications insurance issuers use to manage underwriting, pricing, and policy administration. Neuron connects to insurer systems via APIs, allowing insurers to offer algorithmic responses (Accept, Reject, or Refer) for risk management through the auto-follow feature.
3. Identity and Access Management (IAM) Systems: Neuron integrates with IAM systems to manage user authentication and authorisation for different tenants. This ensures secure access to the Platform's features and functionalities while maintaining data segregation between tenants.

Note: We will be utilising the existing features of MuleSoft for monitoring the usage and availability of the platform.

STAKEHOLDERS

1. Broker Users: Brokers represent the primary users of Neuron, leveraging the Platform to access the marketplace, discover insurers, and manage risk through the auto-follow feature.
2. Support Users: Support users include Neuron's technical support staff and administrators, who are responsible for maintaining the Platform, troubleshooting issues, and ensuring a seamless user experience.
3. Insurer Users: Insurers represent the users of Neuron application. Their responsibility include the tracking and management of the quote responses generated by the algorithm.

The System Context of Neuron highlights its position within the insurance risk management ecosystem, emphasising the importance of seamless integration with external systems and effective collaboration with various stakeholders. By understanding the system's context, developers can ensure that Neuron's architecture is well-suited to fulfil its purpose and deliver user value in a multi-tenant environment.

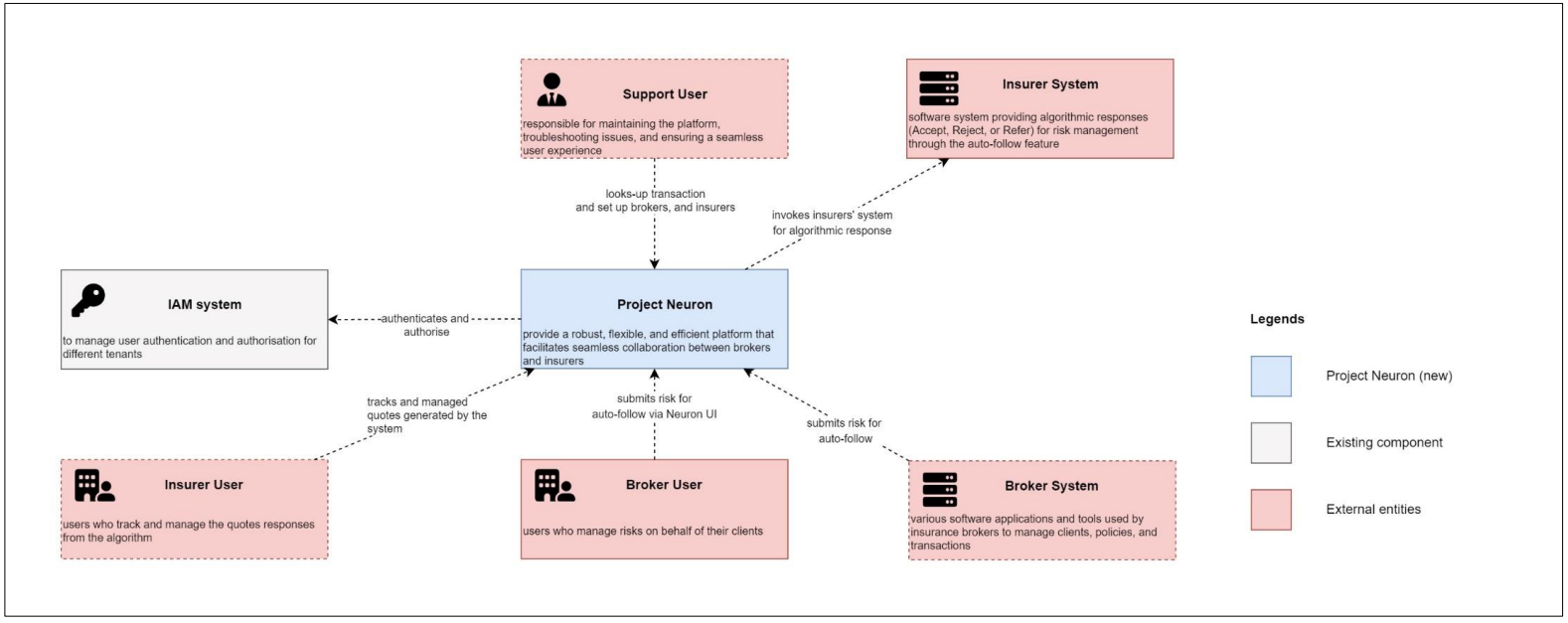

Figure 1 System Context Visualisation

CONTAINER

The Container section of the C4 visualisation provides a high-level view of the main components within the Neuron platform, focusing on the technology choices, major functional areas, and their interactions. This section further decomposes Neuron into its main subsystems, allowing a deeper understanding of the the Platform's architecture.

1. Frontend User Interface: The frontend user interface is the primary touchpoint for insurance brokers and support users. It is a Single Page Application (SPA) built using Angular to provide a seamless and efficient user experience. The frontend user interface communicates with the Neuron Experience API to access the Platform's features and functionalities. It is deployed on Azure with its DNS, ensuring easy access and reliability.
2. Neuron Experience API: The Neuron Experience API is the core component of the Platform, exposing RESTful APIs that enable the frontend user interface and external systems to access the Platform's features and functionalities. The API is designed following the API-first principle, facilitating integration with brokers, insurers, and Azure AD. It is built using MuleSoft's Any Point Platform and deployed on the MuleSoft Cloud Hub for scalability, resilience, and flexibility. The Neuron Experience API is exposed to the internet using its DNS, allowing for secure and efficient communication with the front end and other external systems.
3. Backend Services: The backend services are a collection of microservices responsible for implementing the Platform's business logic. These services follow the microservices architecture pattern, ensuring modularity, scalability, and maintainability. Each service is responsible for a specific functional area within the Platform, such as auto-follow risk management, insurer discovery, or quote management. The backend services are built using MuleSoft's Any Point Platform and deployed on the MuleSoft Cloud Hub. These services are air-gapped from the internet, ensuring increased security and isolation.
4. Data Storage: The data storage layer consists of MongoDB databases hosted on AWS using the Database as a Service model. MongoDB ensures efficient and scalable data storage generated by Neuron's features and functionalities. The databases are designed to provide data segregation for different tenants, maintaining privacy and security. The MongoDB database guarantees data consistency and accuracy across the Platform as the single source of truth.
5. Azure Active Directory (Azure AD) Integration: Neuron integrates with Azure AD using the Multi-tenant Application Model to manage user authentication and authorisation, providing secure access to the Platform's features and functionalities for different tenants. The Neuron Experience API communicates with Azure AD using standard protocols - OpenID Connect.

The Container section of the C4 visualisation helps to identify the main components within the Neuron platform, their technology choices, and their interactions. By understanding the container view, developers and architects can ensure the Platform is built using appropriate technologies and follows best practices for modularity, scalability, and maintainability.

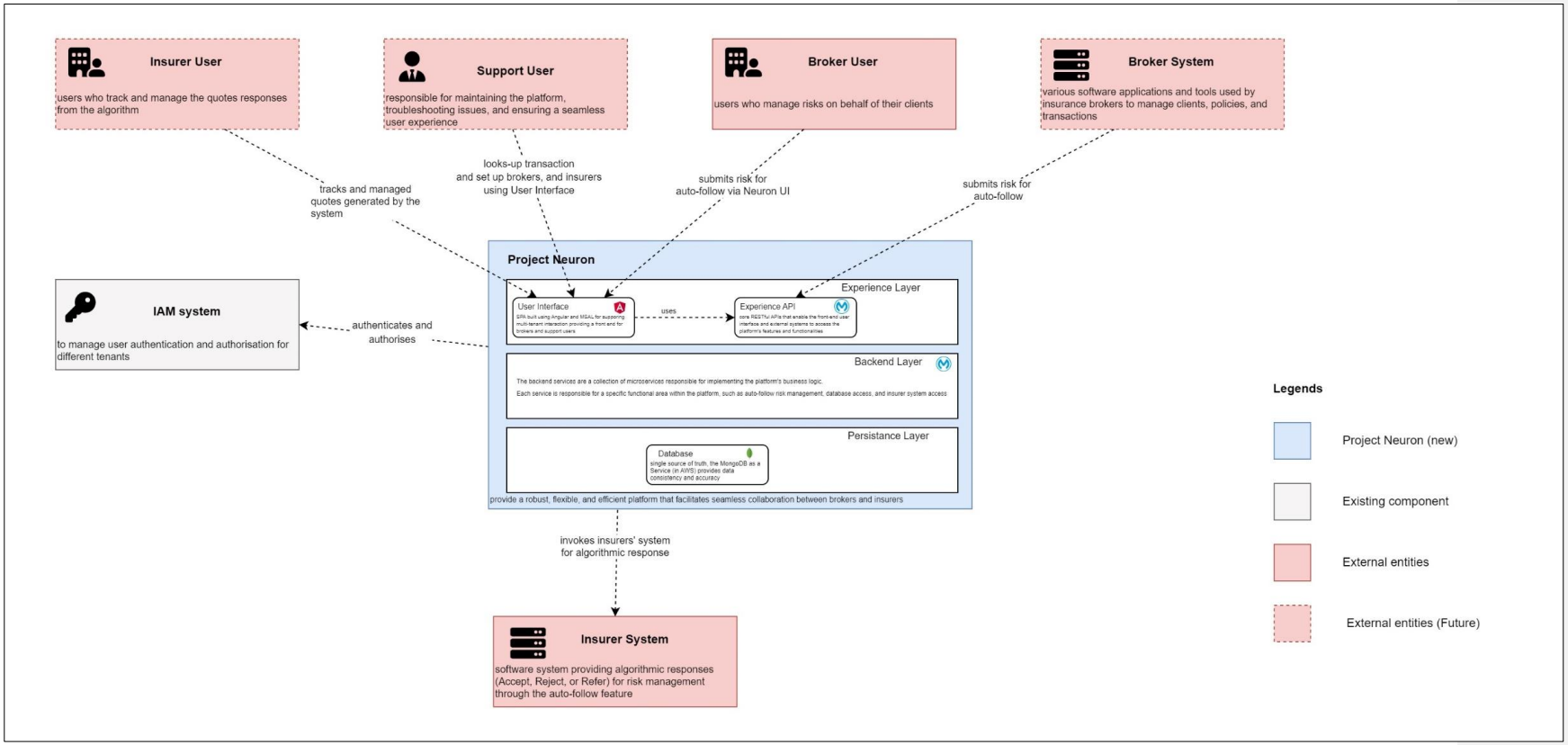

Figure 2 Container Visualisation

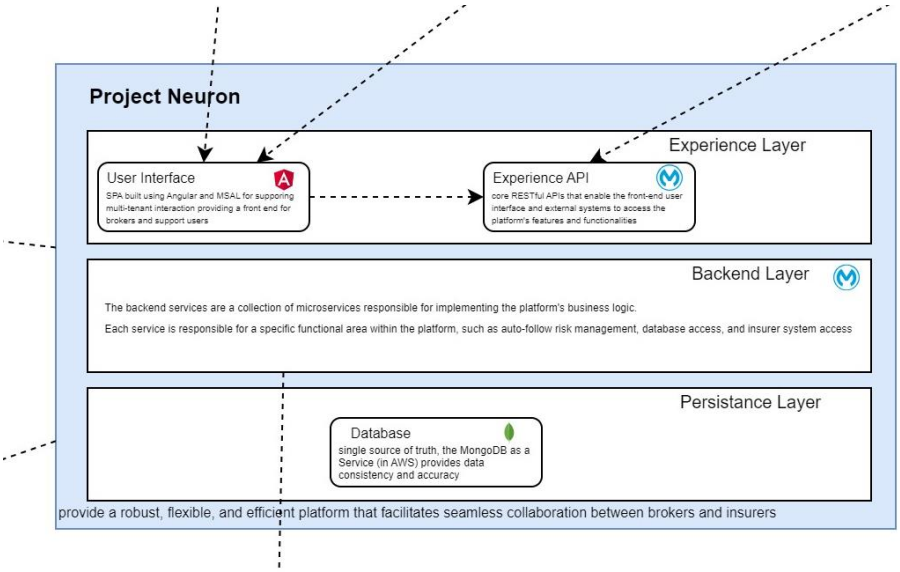

Figure 3 Container Visualisation (Zoomed)

Component

The Component section of the C4 visualisation delves deeper into the Neuron platform, breaking down the main components and their interactions. This level of detail helps developers and architects understand the Platform's architecture at a granular level, offering insights into each component's functional organisation and responsibilities.

1. Frontend User Interface: The frontend user interface remains consistent with the Container section, serving as the primary touchpoint for insurance brokers and support users. It is deployed on Azure with its DNS exposed to the internet, ensuring easy access and reliability.
2. Experience APIs:
    a. Neuron Risk Management API: This Experience API provides an interface for managing risk-related features and functionality, including the auto-follow feature and insurer discovery. It is deployed on MuleSoft Cloud Hub with its DNS exposed to the internet through the MuleSoft Gateway. It incorporates the Dynamic Routing Pattern to route requests to appropriate backend services.
    b. Neuron System Management API: This Experience API manages system-related features and functionality, such as user authentication and authorisation, distribution list management, and data access. It is deployed on MuleSoft Cloud Hub with its DNS exposed to the internet through the MuleSoft Gateway.
3. Process APIs:
    a. Distribution List API: This Process API orchestrates creating and managing distribution lists, enabling brokers to define target groups of insurers for specific risk profiles and streamline the distribution of risk information. It utilises the Guaranteed Delivery Pattern to deliver messages reliably between components.
4. System APIs:
    a. Data Access API: This System API provides a stable and secure interface for accessing and managing data within Neuron's MongoDB data storage layer. It employs the TTL (Time-To-Live) Pattern to manage data expiration and maintain optimal performance.
    b. Insurer Access API: This System API integrates with insurer systems, providing a secure and stable interface for communication between Neuron and insurer systems. The Idempotent Receiver Pattern ensures that duplicate messages are handled correctly and don't cause unintended side effects.
    c. Security Component: This component manages the integration with Azure AD for user authentication and authorisation, ensuring secure access to the Platform's features and functionalities for different tenants.
5. Data Storage: The Data Storage layer remains consistent with the Container section, utilising MongoDB on AWS as the single source of truth for data storage.

The Component section of the C4 visualisation offers a deeper understanding of Neuron's architecture, highlighting each component's functional organisation and responsibilities within the Platform. This detailed view allows developers and architects to ensure the Platform is built using appropriate technologies and follows the best modularity, scalability, and maintainability practices.

Note: This is an evolving architecture; other components can be added as the scope is discovered.

MuleSoft SA will provide the detailed component architecture of MuleSoft Components.

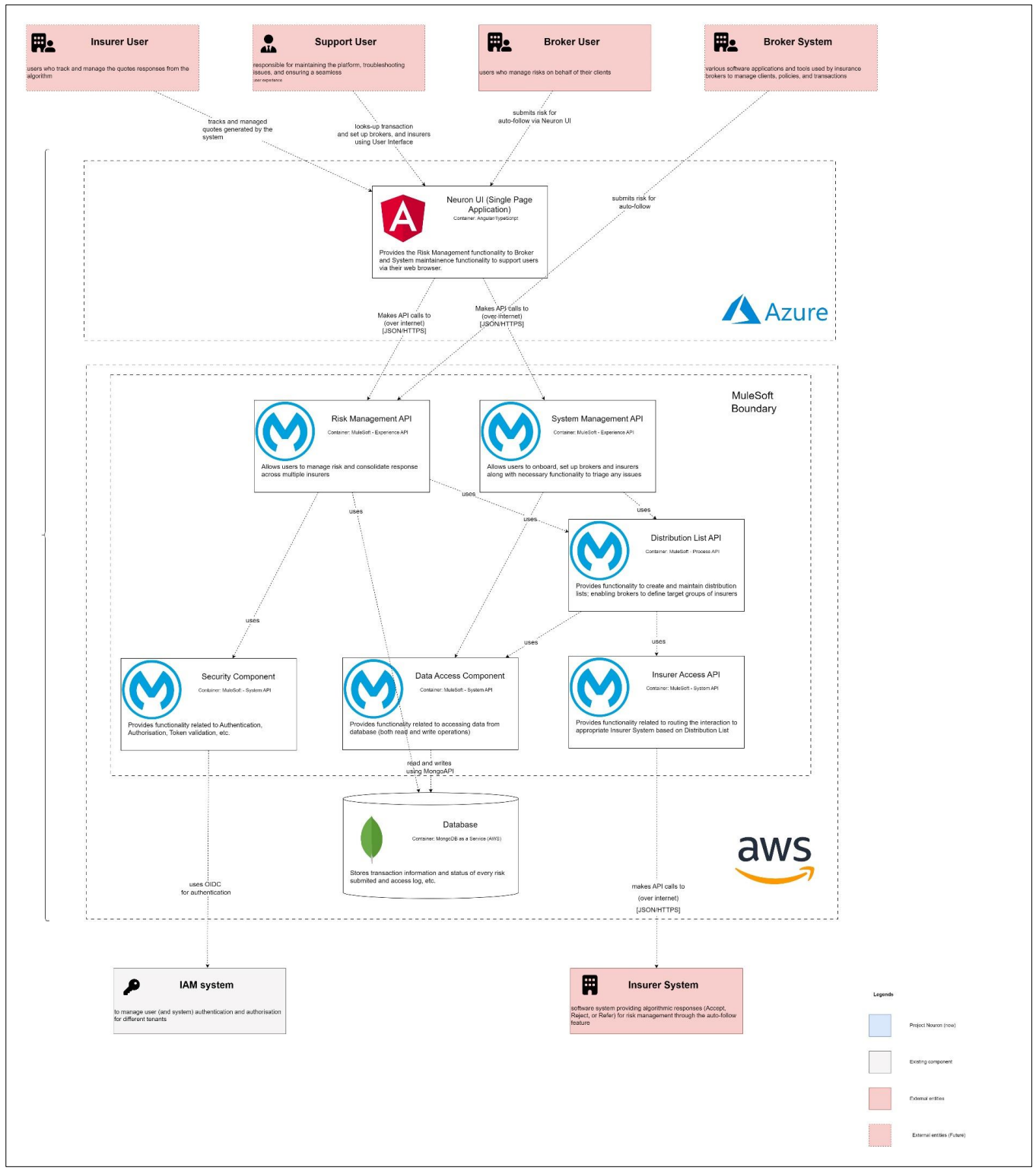

Figure 4 Component Visualisation

NON-FUNCTIONAL REQUIREMENTS

The 'Neuron' platform's non-functional requirements address various aspects of the system's quality attributes, ensuring a robust, scalable, and maintainable solution. Below are the numbers based on the current product projections:

1. Total risks per year: 10,000
2. Working days per year (excluding weekends): 5 days * 52 weeks = 260 days
3. Working hours per day: 8 hours
4. Number of brokers: 4
5. Number of insurers: 15
6. Team members per broker: 10-15

To calculate the TPS requirement, we first need to determine the total number of transactions per day:

```
10,000 risks per year / 260 working days per year = ~38.5 risks per day
```

Next, we calculate the number of transactions per hour:

```
38.5 risks per day / 8 working hours per day = ~4.8 risks per hour
```

Now, let's calculate the TPS:

```
4.8 risks per hour / 3600 seconds per hour = ~0.0013 risks per second or
~0.08 TPS
```

To estimate the concurrent users, we need to consider the number of team members for each broker and insurer:

```
4 brokers * 10-15 team members = 40-60 broker team members
15 insurers * 0 team members = 0 insurer team members
Total concurrent users (assuming 10-20%) = 4-12 concurrent users
```

QUALITY METRICS

Suggested Quality Metrics for the Neuron platform include:

1. Conceptual Integrity: Neuron employs design patterns and architectural styles to ensure a coherent and maintainable system. Metrics such as afferent coupling, efferent coupling, and adherence to design patterns can be used to measure the system's conceptual integrity.
2. Maintainability: Neuron focuses on maintainability by tracking metrics like cyclomatic complexity, type size, percentage of comments, and efferent coupling at the type level to ensure the system remains manageable and extensible.
3. Reusability: The Platform aims to maximise the reusability of components and libraries, focusing on creating a modular and reusable codebase to streamline development efforts.
4. Availability: Neuron strives for high availability by minimising planned downtime and reducing the time required for software and hardware updates on the running system.
5. Interoperability: The Platform supports various integration protocols and standards to facilitate seamless communication with external systems while maintaining backward compatibility for the integration API.

6. Manageability: Neuron ensures the system is easily manageable through comprehensive logging, monitoring, and diagnostic tools that are well-documented and accessible to administrators.
7. Performance: The Platform aims to provide optimal performance by addressing metrics like the number of concurrent users, data storage size and growth, page load times, and function call times.
8. Reliability: Neuron focuses on achieving high reliability by minimising failure rates, optimising mean time to failure (MTTF), mean time to repair (MTTR), mean time between failures (MTBF), and reducing the number of critical and high-severity customer-reported bugs.
9. Scalability: The system's architecture allows for horizontal scaling, with provisions to scale up or down in response to changing demands. Neuron also addresses the scalability of various solution components and ensures that scaling limits are sufficient for the business domain.
10. Security: Neuron prioritises security by addressing PII security scenarios, DDoS attack prevention and mitigation, user authentication and authorisation, SQL injection prevention, and secure handling of sensitive data.
11. Testability: The Platform emphasises testability through comprehensive unit and integration test coverage and a clear list of required test environments and testing approaches.
12. Auditability: Neuron ensures that essential operations leave an audit trail and maintains a record of user activities for auditing purposes.
13. Usability: The Platform follows specific UI/UX guidelines, supports various devices, resolutions, and operating systems, complies with accessibility standards, and prioritises user experience by minimising the number of clicks required to access specific functionalities.

No.	*Quality Attribute*	*Metric*	*Target Value*	*Measurement Method*
1.	Availability	System uptime percentage	99.9%	Monitoring system uptime and downtime, calculating the percentage of time the system is operational
2.	Scalability	Concurrent user capacity	4-12 users	Load testing and system logs to measure the system's ability to scale under different load conditions
3.	Performance	Response time for API transactions	<1 second	Load testing, system logs, and performance monitoring tools
		Function call times	<50ms	Load testing, system logs, and performance monitoring tools
		Page load time	<2 seconds	Browser performance testing tools
		Transactions per second (TPS)	≥0.08 TPS	Load testing, system logs, and performance monitoring tools
4.	Reliability	Mean time to failure (MTTF)	>1 month	System monitoring, error tracking, and bug reporting
		Mean time to repair (MTTR)	<4 hours	System monitoring, error tracking, and bug reporting
		Mean time between failures (MTBF)	>1 month	System monitoring, error tracking, and bug reporting

		Number of high-severity bugs	<1 per month	Bug reporting and tracking
5.	Security	Compliance with industry standards	Yes	Regular security audits and penetration tests
		Number of security vulnerabilities	0	Code analysis for potential vulnerabilities and regular security audits
6.	Testability	Unit test coverage	>80%	Analysing system test coverage, test automation, and test environment availability
		Integration test coverage	>70%	Analysing system test coverage, test automation, and test environment availability
7.	Maintainability	Cyclomatic complexity	<10	Static analysis tools (e.g., SonarQube, Code Climate)
		Type size	<200 lines	Static analysis tools (e.g., SonarQube, Code Climate)
		Percentage of comments	>30%	Static analysis tools (e.g., SonarQube, Code Climate)
8.	Reusability	Number of reusable components	>50%	Code analysis and software design review for identifying and tracking reusable components
9.	Interoperability	Compliance with integration standards	Yes	Integration testing, code analysis, and adherence to integration protocols and standards
10.	Manageability	Comprehensive logging coverage	>90%	Code analysis, system monitoring, and logging tools
		Monitoring and diagnostic tools	Available	System monitoring, log analysis, and diagnostic tool availability
11.	Auditability	Audit trail completeness	>95%	Code analysis, log analysis, and audit trail inspection
12.	Usability	Compliance with UI/UX guidelines	Yes	User testing, user feedback, and adherence to UI/UX guidelines
		Support for various devices	Yes	Cross-device and cross-browser testing. Supported browsers: • Google Chrome (latest version) • Mozilla Firefox (latest version) • Microsoft Edge (latest version) • Apple Safari (latest version)
		Accessibility compliance	Yes	Accessibility testing and adherence to accessibility standards (e.g., WCAG 2.0)
		Number of clicks for specific features	Minimal	User testing, user feedback, and user interface analysis
		Locale Supported	EN-GB	
13.	Conceptual Integrity	Design Patterns: 1. Guaranteed Delivery Design Pattern 2. Time To Live Design Patten	Yes	Design review and analysis.

3. Idempotent Receiver Pattern
4. Dynamic Router Pattern

Note: Some of the metrics are not directly applicable to the Any Point Platform – they need to be translated to Any Point Studio. Please refer to the solution documentation for more information.

Information Architecture

The Information Architecture in the Neuron platform is designed to ensure efficient and secure data handling at every stage of its lifecycle. The following components are used:

1. Data collection: Data is collected from various sources, such as external APIs and monitoring endpoints. Data is compiled using the Any Point Platform.
2. Data storage: Data is stored in MongoDB databases hosted on AWS using the Database as a Service model. MongoDB ensures efficient and scalable data storage generated by Neuron's features and functionalities. The databases are designed to ensure data segregation for different tenants, maintaining privacy and security. MongoDB also provides built-in support for data archiving, which can be configured to automatically move data to a separate storage tier based on specific retention policies.
3. Data processing: Various system components processes data, such as the Neuron Experience API and Radar Live SaaS. Data is transformed, validated, and enriched to ensure accuracy and consistency. Data processing is performed within the system, with no external systems accessing the data.
4. Data sharing: Data is shared with external systems and third-party services through APIs and secure channels. The Neuron Experience API exposes RESTful APIs that enable frontend user interface and external systems to access the Platform’s features and functionalities. The APIs are designed following the API-first principle, facilitating integration with brokers, insurers, and Azure AD.
   To ensure confidentiality and integrity, data is protected using encryption, access controls, and authentication using OpenID Connect.
5. Data archiving: Data is archived for compliance and long-term retention within the MongoDB databases. The archiving process is automated and can be configured to move data to different storage tiers based on specific retention policies. Archived data can be accessed for audit and compliance purposes.

It's important to note that data validation and quality checks are performed at various stages of the data flow to ensure accuracy and consistency. Additionally, data is secured using TLS 1.3 encryption and access controls to protect sensitive information. The Information Architecture in the Neuron platform ensures efficient and secure data handling throughout its lifecycle, providing a reliable and scalable platform for insurance brokers and insurers.

Deployment Architecture

Neuron's Deployment Architecture is designed to provide high availability, scalability, and resilience. The system is deployed on Azure and AWS, combining Platform as a Service (PaaS) and Software as a Service (SaaS) offerings frontend user interface, built using Angular, is deployed as a Single Page Application (SPA) on Azure, with its DNS exposed to the internet. This deployment approach ensures easy access and reliability for insurance brokers and support users.

All APIs are deployed on MuleSoft's Cloud Hub, a SaaS offering that provides a scalable, resilient, and flexible integration platform. Cloud Hub runs on top of AWS, leveraging its advanced infrastructure and security features. The Neuron Experience API, Risk Management API, System Management API, Distribution List API, Insurer Access API, and Data Access API are deployed as MuleSoft applications, each with its DNS. The APIs are exposed to the internet using the MuleSoft Gateway, ensuring secure and efficient communication with the front end and other external systems.

Neuron leverages the multi-tenant Azure Active Directory (AAD) to manage user authentication and authorisation, providing secure access to the Platform’s features and functionalities for different tenants. The Control Plane uses WTW's AD, which is only accessible to internal developers.

The deployment architecture ensures high availability, scalability, and resilience by utilising auto-scaling, load balancing, and disaster recovery capabilities provided by Azure, AWS, and MuleSoft. TLS 1.3 encryption is used for secure communication between different system components.

The deployment architecture is designed to provide easy maintainability and upgradeability, and the system components can be updated and upgraded independently without impacting other components or system availability. The deployment architecture is also designed to ensure easy troubleshooting and monitoring, with system logs collected and monitored by 3rd party tools.

Overall, the deployment architecture is designed to ensure the Neuron platform's reliability, performance, and security.

Typical Deployment Landscape

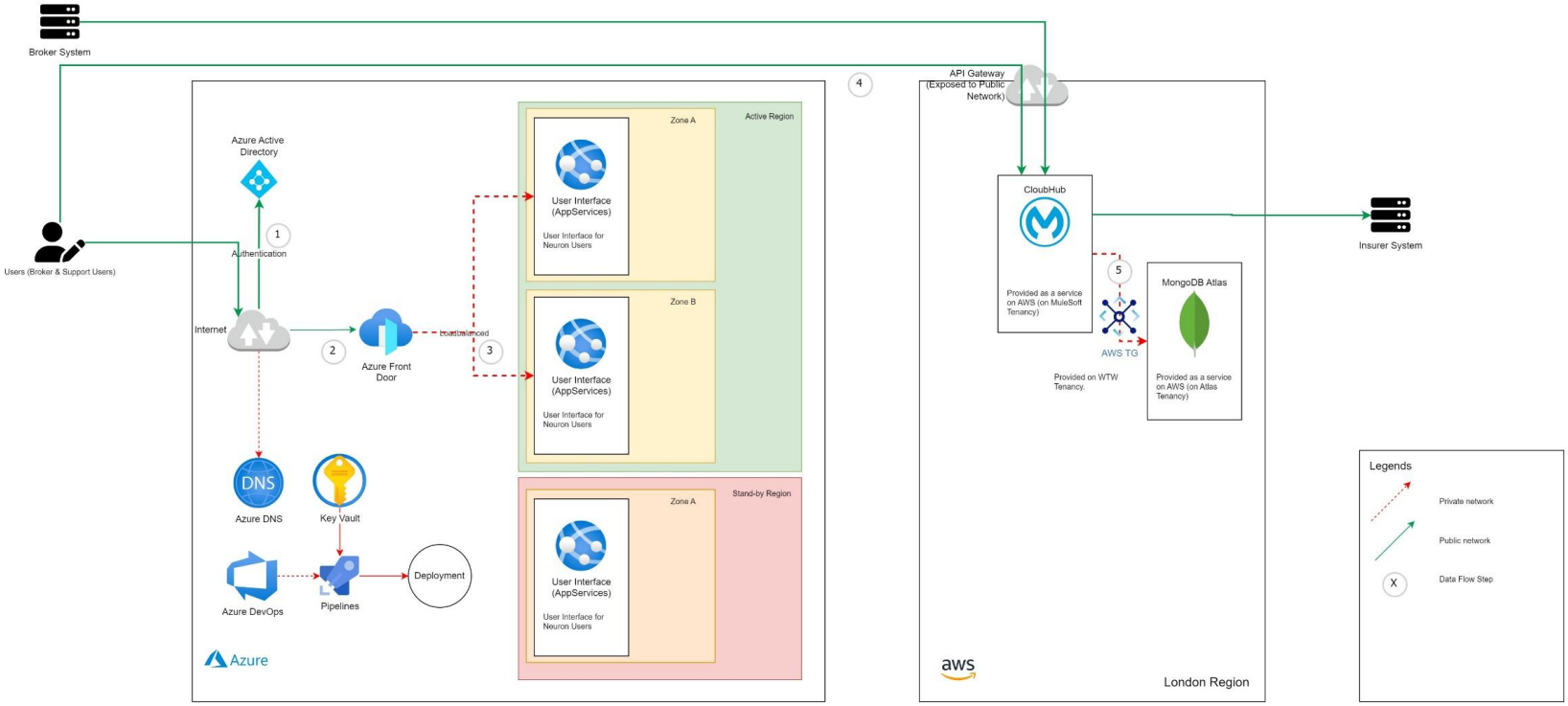

Figure 5 Deployment Landscape

MULTI-ENVIRONMENT STRATEGY

Neuron is deployed using a ***multi-environment strategy*** that includes Development, Test, Pre-Production, and Production environments. The environments are designed to provide a controlled and secure deployment process, with each environment serving a specific purpose in the software development lifecycle.

Development Environment

The Development environment is where Neuron is developed and tested by the development team. The environment is deployed on local workstations or in a private cloud to provide a controlled environment for development and testing.

Test Environment

Developers conduct integration and functional testing in the Test environment, and quality assurance engineers perform acceptance testing. This environment is similar to the Production environment but with smaller capacity and limited data.

Pre-Production Environment

The Pre-Production environment, or the Staging environment, replicates the Production environment. This environment is used for the final testing and validation of software releases before they are promoted to the Production environment.

Production Environment

The Production environment is the live environment where Neuron is available to users. This environment is designed to be highly available, scalable, and resilient. The Production environment is deployed on Azure for SPA (Angular) deployment, and all APIs are deployed in CloudHub, a SaaS offering from MuleSoft, on top of AWS.

Sandbox Environment

Neuron also includes SandBox environments for Broker and Insurer Integration. The Sandbox environment is designed to provide a safe and isolated environment for testing and development by external stakeholders, such as brokers and insurers, without affecting the live production environment. The SandBox environment is deployed on Azure for SPA (Angular) deployment, and all APIs are deployed in CloudHub on top of AWS, similar to the Production environment.

Overall, the deployment architecture of Neuron is designed to ensure a reliable and scalable deployment process, providing secure environments for development, testing, and production use and a safe environment for external stakeholders to test and develop with Neuron.

DISASTER RECOVERY & BCP

Neuron's deployment architecture is designed to ensure high availability, scalability, and resilience through the utilisation of auto-scaling, load balancing, and disaster recovery capabilities provided by Azure, AWS, and MuleSoft. This architecture is designed to provide a reliable and scalable deployment process that ensures the security and stability of the system.

MuleSoft provides a Disaster Recovery plan for CloudHub that includes replication of data and applications across multiple availability zones within a region. This architecture offers redundancy in an outage in a single availability zone. In the case of a complete region failure, MuleSoft's Disaster Recovery plan includes replicating data and applications across multiple regions.

MuleSoft Solution Architect will come up with a Disaster Recovery plan and Business Continuity Plan (BCP) that outlines the recovery time objectives (RTO) and recovery point objectives (RPO) for Neuron. The Disaster Recovery plan will guide the actions that need to be taken in the event of a disaster, such as a natural disaster, power outage, or cyber attack.

The BCP will define the policies and procedures to ensure the continued operation of "Neuron "during an emergency or disruption. The BCP will guide the actions that need to be taken to restore services and ensure the continuity of operations. The BCP will also outline the roles and responsibilities of the stakeholders involved in the disaster recovery process.

The Disaster Recovery Plan and BCP will be regularly tested and updated to ensure their effectiveness during a disaster. Testing the Disaster Recovery plan will involve simulating a disaster to determine the effectiveness of the recovery procedures. The BCP will be tested through tabletop exercises that simulate various disaster scenarios and test the stakeholders' responses.

The Disaster Recovery Plan and BCP are critical components of Neuron's deployment architecture. MuleSoft Solution Architect will work closely with the Neuron team to ensure the system remains highly available, scalable, and resilient, even during a disaster.

SECURITY

The Security section of Neuron encompasses measures to safeguard the Platform's integrity, confidentiality, and availability, ensuring that users' data and assets are protected from malicious actors. It includes the following components:

1. Public Documentation Portal: The Public Documentation Portal is a component of Neuron that allows users to access the Platform's documentation and related resources without any authentication. It is deployed on top of the AnyPoint Exchange, a service provided by MuleSoft. The portal is exposed to the internet, and measures are taken to ensure that only authorised documentation and resources are available to the public.
2. Control Plane: The Control Plane is a management plane that controls the infrastructure of the Neuron platform, including the deployment of components, scaling, monitoring, and security. It is hosted on MuleSoft's US Control Plane. WTW's Azure Active Directory is used for authentication and authorisation, ensuring that only authorised personnel can access and manage the Platform's infrastructure.
3. Runtime Plane: The Runtime Plane is the execution environment for Neuron's backend services, including the Neuron Experience API and System and Process APIs. It is hosted on the MuleSoft CloudHub, a multi-tenant platform, and uses Azure Active Directory (AAD) for identity and access management. AAD ensures that each tenant's data is segregated and secure, and only authorised users can access the Platform's features and functionalities.

In addition to the above components, the following measures are taken to ensure the Platform's security:

1. Data encryption: Data in transit and at rest is encrypted using industry-standard protocols to prevent unauthorised access and protect users' sensitive information.
2. Firewall and Access Control: Neuron employs strict firewall and access control policies to protect the Platform from unauthorised access, prevent data breaches, and ensure the Platform's availability.
3. Regular Audits: Regular audits and penetration testing are conducted to identify and address any security vulnerabilities or threats to the Platform.

By implementing the above measures, Neuron ensures that its users' data and assets are protected from unauthorised access and malicious actors, providing a secure and reliable insurance risk management platform.

AUTHENTICATION

User Authentication

The user authentication process within the Neuron platform is designed to offer a secure and seamless experience for users accessing the system. The platform leverages Azure Active Directory (AAD) for multi-tenant single sign-on (SSO), enabling users from various organisations to authenticate and access the platform with their respective AAD credentials.

Upon attempting to access the Neuron platform, the user is redirected to AAD using MSAL.js. AAD shows the user a tenant-specific login page where they provide their credentials. AAD handles the OIDC authentication

and, upon successful authentication, returns an ID token and access token to the Angular Single Page Application (SPA).

The user can now interact with the SPA, which sends API requests to the Neuron API (MuleSoft) and the AAD access token. The Neuron API validates the access token with AAD. If the token is valid, the API processes the request and returns a response to the SPA. If the token is invalid or expired, the API returns an HTTP status code of 401 (Unauthorized), and the user is shown a 401 error message.

In case of unsuccessful authentication by AAD, an error message is returned, and the SPA displays this error to the user.

In summary, the user authentication process within the Neuron platform leverages the security and flexibility of AAD to provide a secure and efficient platform for collaboration and risk management between brokers and insurers. MSAL.js for redirecting users to AAD for authentication fully utilises AAD's multi-tenant functionality.

Sequence Diagram

```
@startuml
actor User
participant "Angular SPA" as SPA
participant "Azure AD" as AAD
participant "Neuron API (MuleSoft)" as MuleAPI

User -> SPA: Access SPA
SPA -> AAD: Redirect to AAD using MSAL.js
AAD -> User: Show tenant-specific login page
User -> AAD: Provide tenant-specific credentials
alt Successful Authentication
    AAD -> SPA: Return ID token and access token
    User -> SPA: Interact with the application
    SPA -> MuleAPI: Send API request with the access token
    MuleAPI -> AAD: Validate access token
    AAD -> MuleAPI: Return token validation result
    alt Token is valid
        MuleAPI -> SPA: Process request and return response
    else Token is invalid or expired
        MuleAPI -> SPA: Return 401 error (Unauthorised)
        SPA -> User: Show 401 error message
    end
else Unsuccessful Authentication
    AAD -> SPA: Return error message
    SPA -> User: Show error message
end
@enduml
```

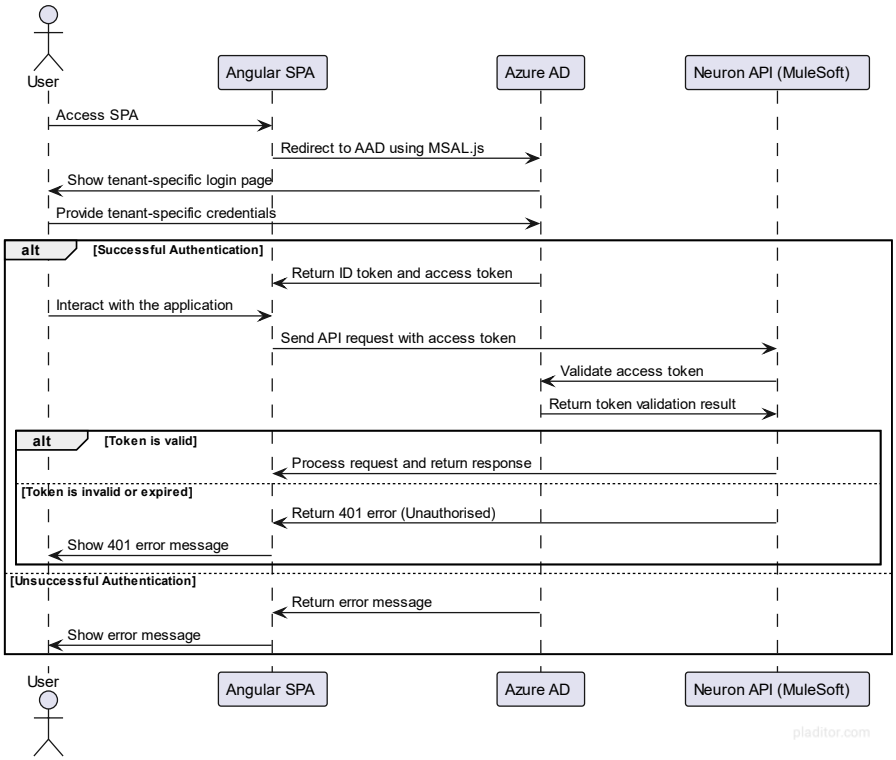

Figure 6 Neuron UI Authentication Sequence Diagram

Broker System Authentication

The Broker System Authentication within the Neuron platform uses OpenID Connect (OIDC) on Azure AD with multi-tenant support. This method allows broker systems to bring their identities for authentication, thus enabling secure interactions with the platform. Individual users can also be onboarded as guests onto Neuron's Azure AD as a fallback authentication method.

When a broker system tries to access the Neuron platform, it initiates a “client_credentials grant flow” with Azure AD. Azure AD validates the provided “client_id” and “client_secret” from the broker system. Azure AD issues an access token to the broker system if the validation is successful.

Armed with the access token, the broker system sends API requests to the Neuron API (MuleSoft). The Neuron API validates this access token with Azure AD. If the token is valid, the API processes the request and returns a response to the broker system. If the token is invalid or expired, the API returns an HTTP status code 401 (Unauthorized).

In the case of unsuccessful validation by Azure AD, an error message is returned to the broker system.

The updated authentication approach leverages the security and scalability of Azure AD and OIDC, ensuring a secure, seamless experience while aligning with the latest industry standards and best practices.

Sequence Diagram

```
@startuml
actor "Broker System" as BrokerSys
participant "Azure AD" as AAD
participant "Neuron API (MuleSoft)" as MuleAPI

BrokerSys -> AAD: Initiate client_credentials grant flow
AAD -> BrokerSys: Validate client_id and client_secret
alt Successful Validation
    AAD -> BrokerSys: Return access token
    BrokerSys -> MuleAPI: Send API request with the access token
    MuleAPI -> AAD: Validate access token
    AAD -> MuleAPI: Return token validation result
    alt Token is valid
        MuleAPI -> BrokerSys: Process request and return response
    else Token is invalid or expired
        MuleAPI -> BrokerSys: Return 401 error (Unauthorised)
    end
else Unsuccessful Validation
    AAD -> BrokerSys: Return error message
end
@enduml
```

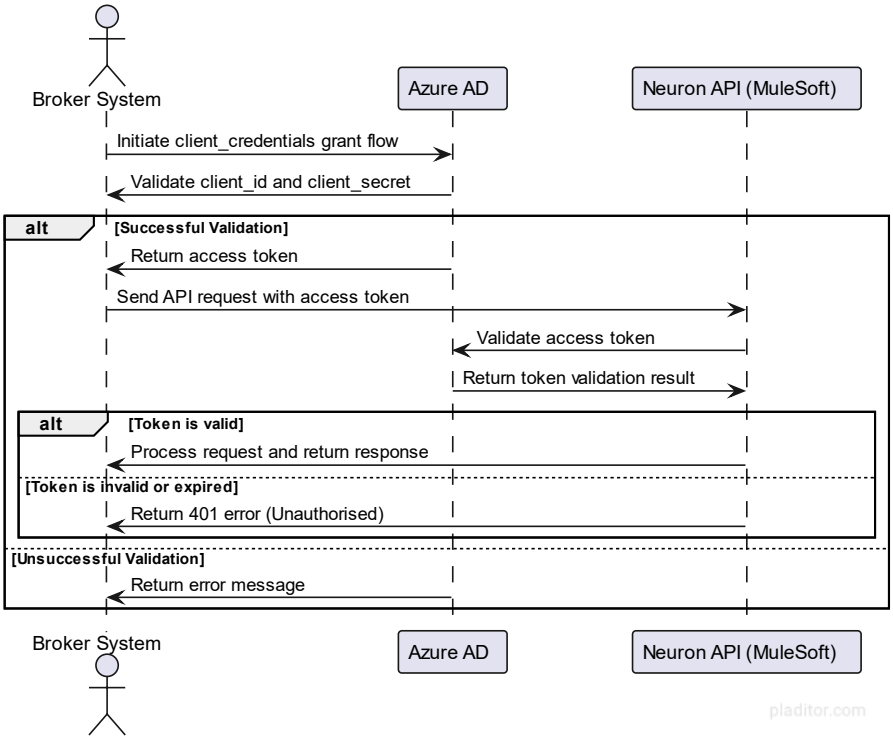

Figure 7 Broker System Authentication Sequence Diagram

AUTHORISATION

Authorisation in MuleSoft-led API design involves defining access control policies to restrict or permit access to specific resources or actions. This is typically achieved by implementing security policies that enforce authorisation rules based on user roles, permissions, or other attributes.

MuleSoft provides various security policies that can be applied to APIs to enforce authorisation rules, such as OAuth2, Basic Authentication, or Mutual TLS. These policies can be configured to authenticate and authorise requests based on specific user roles, groups, or the request's characteristics.

Working Assumption 1: The current authorisation approach assumes roles allocated through Azure AD's AppRoles for managing user access. The IAM admin assigns these roles, which are then used for authorising access to different API resources.

Working Assumption 2: JWT Policy Validation is used for enforcing authorisation within the MuleSoft platform. In addition, the role of permission configuration will be automated through a CI-CD pipeline based on a predefined configuration.

Pros of using AppRoles and JWT Policy Validation

1. Centralised role management through Azure AD simplifies administration and improves security.
2. Role-based access control (RBAC) provides granular control over API access.

3. Integration with Azure AD enables seamless user authentication and authorisation.
4. App Roles can be easily maintained, updated, or modified as business requirements change.
5. JWT Policy Validation ensures secure and efficient validation of access tokens.
6. Automated CI-CD pipeline for role-to-permission configuration reduces manual intervention and potential errors.

Cons of using App Roles and JWT Policy Validation

1. Dependency on Azure AD for role management could concern organisations that use something other than Azure AD.
2. App Roles may need to provide the desired level of granularity for more complex access control scenarios.
3. JWT Policy Validation may require additional configuration and management efforts.
4.

The specific policies and authorisation rules to be implemented will depend on the requirements and context of each API. The MuleSoft solution architect, in consultation with stakeholders and subject matter experts, will determine them. The policies will ensure the appropriate access control and data protection level for each API, and they will be documented in detail as part of the API design specification.

```
@startuml
actor User
participant "Angular SPA" as SPA
participant "Azure AD" as AAD
participant "Neuron API" as API

User -> SPA: Access Neuron SPA
SPA -> AAD: Redirect to Azure AD login
User -> AAD: Provide credentials
AAD -> User: Issue JWT token with embedded roles
User -> SPA: Access Neuron SPA with JWT token
SPA -> API: Request resources with JWT token
API -> AAD: Validate JWT token
AAD -> API: Token validation response
API -> API: Extract roles from the JWT token
alt User has required roles
  API -> SPA: Grant access to resources
else User lacks required roles
  API -> SPA: Deny access to resources (401 or 403)
end
@enduml
```

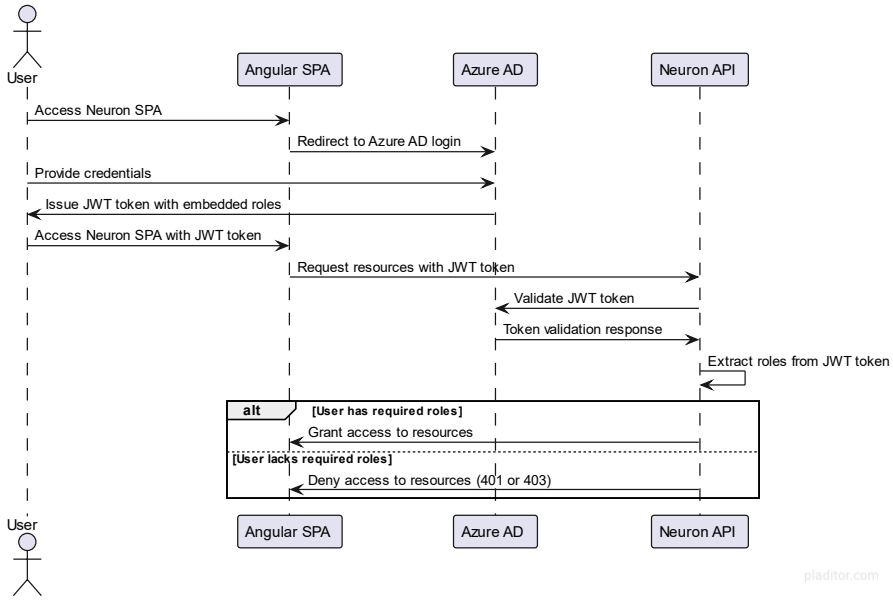

Figure 8 Authorisation

User Authentication and Authorization with Azure AD (Multi-tenancy) and OIDC involve the following steps:

1. When users attempt to access the Neuron platform, they are redirected to Azure AD's login page to authenticate.
2. After providing valid credentials, Azure AD issues a JWT token that includes the user's roles as claims within the token.
3. The Neuron platform validates the JWT token and extracts the roles from the token to determine if the user can access the requested resources or actions.
4. If users have the required roles, they are granted access to the platform and its features based on their assigned permissions and roles.

DATA PRIVACY

Neuron takes data privacy and security seriously, especially involving personally identifiable information (PII) in the Platform's operations. Neuron has implemented a comprehensive set of data privacy measures to ensure compliance with relevant data protection regulations and safeguard user data.

Firstly, the Neuron collects, stores, and processes only the minimum necessary PII data required for its intended purpose. The Platform's data storage is secure and encrypted, with access restricted to authorised personnel.

Neuron follows strict data retention policies, ensuring that PII data is only stored for the purposes it was collected. Regular audits are performed to identify and delete any data that is no longer needed. In addition, Neuron incorporates a consent management system, allowing users to provide, withdraw, or modify their consent for data collection, processing, and sharing following applicable data protection regulations.

The Platform implements robust access control mechanisms, such as role-based access control and multi-factor authentication, to restrict access to PII data and ensure that only authorised users can access the information. All PII data transmitted or stored within the Neuron platform is encrypted, both in transit and at rest, using industry-standard encryption algorithms and protocols.

Neuron also ensures that PII data is only shared with authorised third parties and only when necessary for the intended purpose. Any third-party integrations are vetted to comply with data protection standards and privacy regulations. In case of a data breach, the Platform has a comprehensive data breach detection and response plan in place, including monitoring for potential threats, regular security audits, and incident response procedures to mitigate the impact of a data breach and comply with regulatory notification requirements.

Privacy by design is a core principle of the Neuron platform, with data privacy considerations embedded in every stage of development, from initial design to implementation and maintenance. Regular training and awareness programs are provided to all employees and contractors involved in the Platform's development, operation, and maintenance to ensure they understand the importance of data privacy and the best practices for protecting PII information.

By implementing these measures, Neuron ensures the highest level of data privacy and protection for its users, safeguarding PII information and maintaining compliance with relevant data protection regulations.

APPENDIX

DECISION LOG

Please refer to https://dev.azure.com/rcss-willistowerswatson/Neuron/_wiki/wikis/Neuron.wiki/13020/Decision-Log for the latest updates and decisions.

TABLE OF FIGURES