**Cloud Based Web Application for QCAA -**

**Integrating Aviation Safety and Regulation**

**Business Proposal Document**

Aug 28th 2024

**Prepared for : Infotech Solution, Qatar**

**Prepared By**



**75way Technologies Private Limited**

***Email :*** [*Lokesh@75way.com*](mailto:Lokesh@75way.com)

***Phone*** *: +91 88022 04052*

***Skype*** *: Lbansal2099*

***Telegram*** *: Lokesh75way*

***Address*** *: Plot No. F-335 (P), 75way Tower, Phase 8B, Industrial Area, Sector 74, Mohali, SAS Nagar, Punjab, 160055*

**Table of Content**

[**1. PROJECT OVERVIEW 2**](#_ck9frba541nn)

[**2. OUR GOAL 3**](#_9ev44vqez9lj)

[**3. OBJECTIVE 4**](#_j9bglltzfw07)

[**4. CURRENT ARCHITECTURE 5**](#_dya29ueg33z)

[**4.1 CURRENT TECHNOLOGY STACK 6**](#_aaqruetm4x8p)

[**5. PROPOSED NEW ARCHITECTURE 9**](#_vkea7ptxwx60)

[**5.1 DETAIL OVERVIEW 9**](#_1ddsoct6t9f)

[**5.2 PROPOSED NEW TECHNOLOGY STACK 11**](#_9wx6nv16mn0k)

[**5.3 TECH STACK COMPARISON: Rationale for Upgrading 13**](#_1duotmt731pf)

[**6. INFRASTRUCTURE 15**](#_g0ije2x5re73)

[**7. REQUIRED SYSTEM CONFIG 19**](#_2fap2f4yukqb)

[**8. TECHNICAL SPECIFICATION - SCOPE OF WORK 20**](#_i9r8f2mdkkgk)

[**8.1. Safety Web - Microservice 20**](#_5uanu21zsc4h)

[**8.2. Legitrack - Microservice 23**](#_cnu53vithamj)

[**8.3. AuditTool - Microservice 25**](#_hptvvji7isgz)

[**8.4. Enterprise Architect - Microservice 27**](#_6qu0pvaxffbq)

[**8.5. Workflow - Microservice 27**](#_8ke38me84ap)

[**8.6. Investigation - Microservice 28**](#_vie98n5xyv35)

[**9. TEAM STRUCTURE 30**](#_fipubdbsn1sn)

[**10. APPLICATION TRANSITION AND IMPLEMENTATION PLAN 30**](#_f2frpg5mkx9z)

[**11. TIME & COST ESTIMATION 34**](#_ngs1nagyc0vl)

[**12. MAINTENANCE & SUPPORT TERMS 34**](#_8con1196gclm)

# 

# **1. PROJECT OVERVIEW**

The Qatar Civil Aviation Authority (QCAA) is responsible for ensuring the safety, regulation, and management of Qatar's civil aviation sector. To support its mission, QCAA utilizes a range of specialized web applications designed to manage critical processes such as rulemaking, audit preparation, procedure design, incident reporting, and accident investigation. These applications include:

* **Legitrack**: Facilitates the rulemaking process.
* **Audittool**: Assists in preparation for ICAO audits and Corrective Action Plans (CAP).
* **Enterprise Architect**: Supports the design of procedures and processes.
* **Safetyweb**: Aids in incident reporting and analysis.
* **Workflow**: Facilitates the execution of procedures.
* **Interface**: Creates links between the other applications.
* **Investigation Management**: Supports the investigation of aviation accidents.

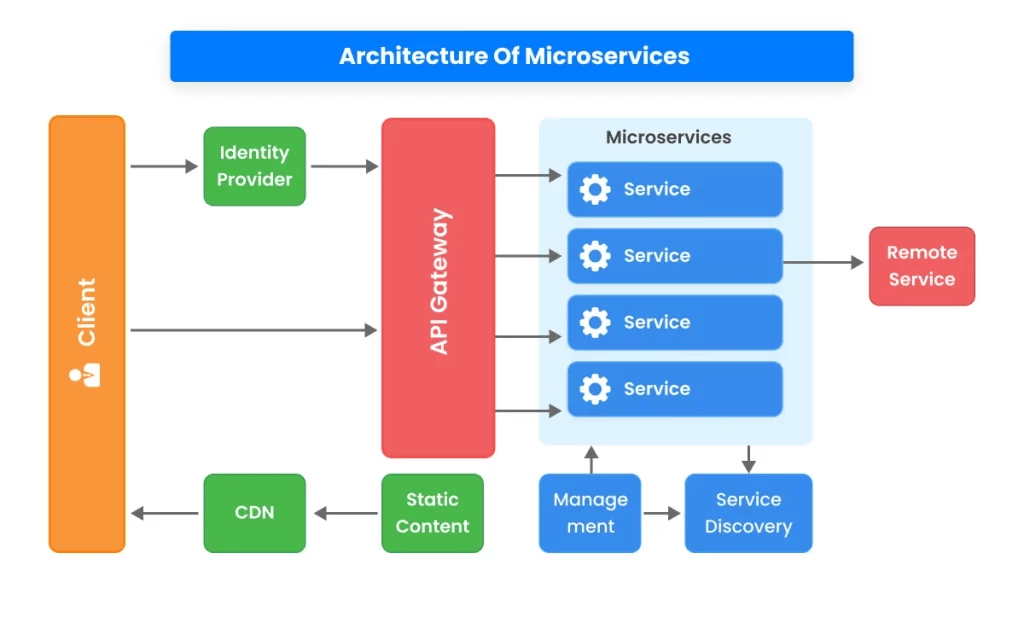
# **2. OUR GOAL**

* To modernize its digital infrastructure, QCAA has decided to transition to a cloud-based architecture using AWS Services /Microsoft Azure. This project proposes the development of a new cloud-based software application system that utilizes a microservices architecture to unify these various applications into a cohesive platform.
* Our goal is to re-construct and develop all the software application system from scratch with a cloud based driven architecture focusing on the latest technology, scalability, better performance & robustness specifically for QCAA. This system will unify the existing QCAA applications into a single, integrated platform designed to streamline processes related to aviation safety and regulations.

# **3. OBJECTIVE**

Key objectives include :

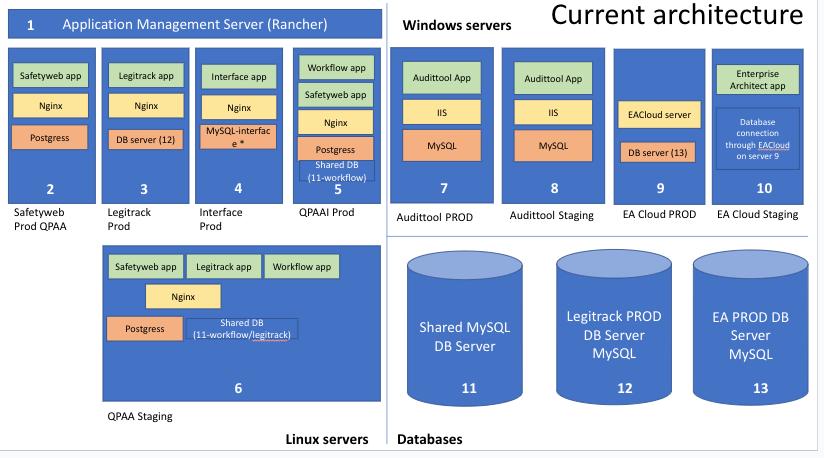
1. **Cloud-Based Microservices Architecture**: Design and implement a cloud-based system on AWS/Microsoft Azure, using a microservices architecture. Each application module—Legitrack, Audittool, Enterprise Architect, Safetyweb, Workflow, Interface, and Investigation Management—will function independently while integrating seamlessly within the unified system. This approach will ensure flexibility, scalability, and ease of maintenance.
2. **Unified System Functionality**: Ensure that all modules work together as a unified system, providing a seamless user experience for QCAA personnel. The system will enable efficient incident reporting, tracking, and management while supporting rulemaking, audits, procedure design, and accident investigations.
3. **Cloud Deployment on AWS/Microsoft Azure**: Transition the hosting of all applications and data to Microsoft Azure's cloud infrastructure. This will provide QCAA with enhanced scalability, reliability, and security, enabling better management of their digital assets.
4. **Scalability and High Availability**: Develop the system to be highly scalable and resilient, capable of handling increased demand and ensuring continuous operation with minimal downtime.
5. **Data Integrity and Security**: Architect the system to maintain the integrity and security of QCAA data, ensuring that all applications are hosted on dedicated cloud resources while allowing for necessary integration and communication between modules.

****

**Upon completion of this project**, QCAA will have a robust, secure, and scalable cloud-based system that enhances its ability to manage aviation safety and regulatory processes, supporting Qatar’s commitment to excellence in civil aviation.

# **4. CURRENT ARCHITECTURE**

Based on our detailed review on the basis of provided material, we have prepared a diagrammatic representation for the current architecture. Attached the diagram :



# 

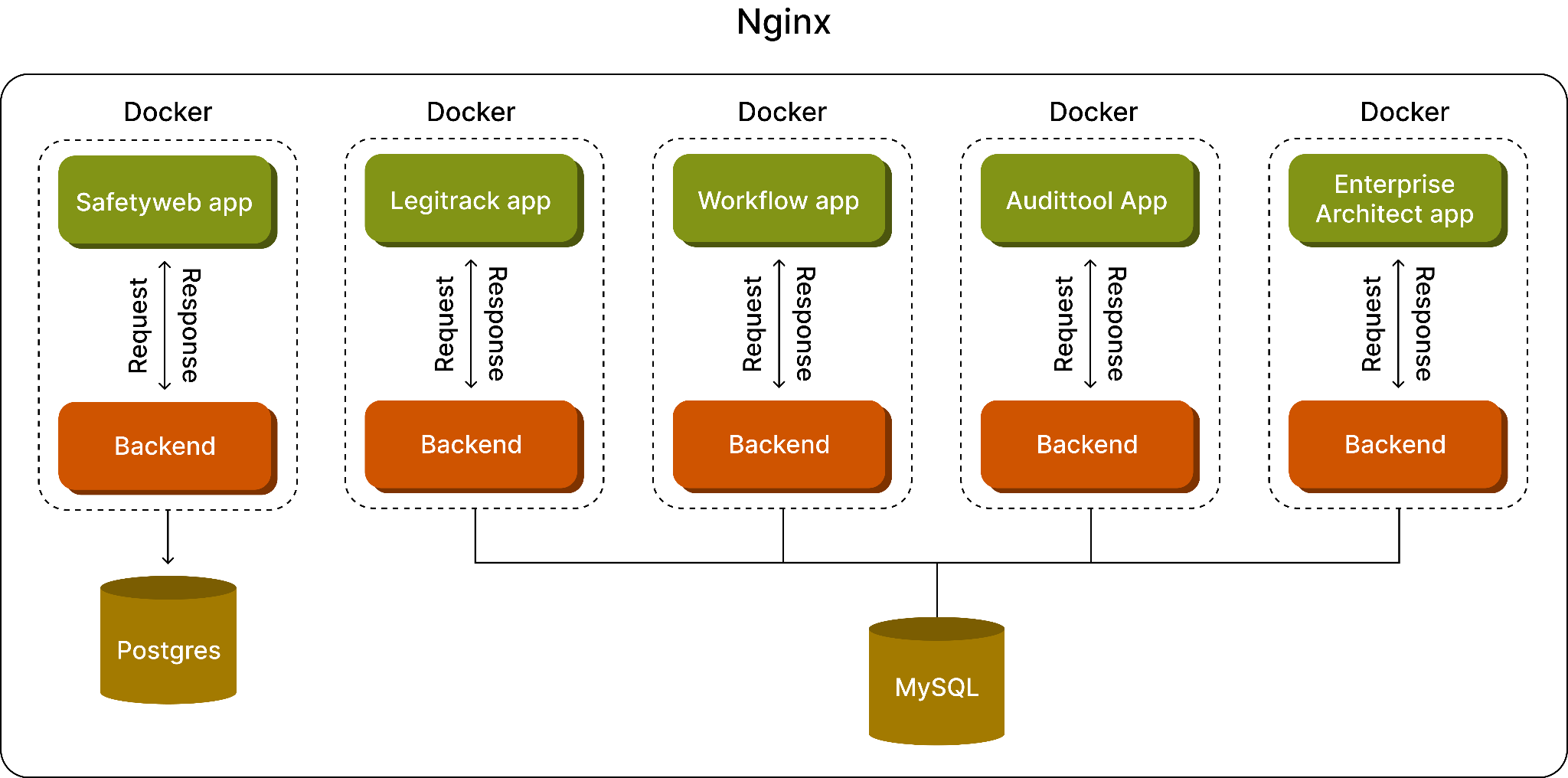
## **4.1 CURRENT TECHNOLOGY STACK**

Following table summarizes the current technology architecture used to build the existing QCAA system :

| **Component** | **Description** | **Technology Stack** | **Database** |
| --- | --- | --- | --- |
| **Application Management Server** | Rancher is used for managing containerized applications' deployment, scaling, and operations across different environments. | Rancher |  |
| **Safetyweb Prod QCAA** | Hosts the Safetyweb app using Nginx as the web server. Managed under the application management server (Rancher) for production. | Nginx, Python, Django | PostgreSQL |
| **Legitrack Prod** | Hosts the Legitrack app with Nginx as the web server. Connects to a separate database server (DB server ). | Nginx, Python, Django | MySQL (DB server) |
| **Interface Prod** | Hosts the Interface app with Nginx as the web server. Connects to a dedicated MySQL database for interface-related data. | Nginx, Python, Django | MySQL (MySQL-interface) |
| **QCAA PROD** | Hosts multiple applications, including Workflow and Safetyweb apps, using Nginx. Shares a PostgreSQL database ("Shared DB 11-workflow"). | Nginx, Python, Django | PostgreSQL ("Shared DB -workflow") |
| **QCAA Staging (Linux Servers)** | A staging environment for Safetyweb, Legitrack, and Workflow apps on Linux servers with Nginx. Uses a shared PostgreSQL database for staging. | Nginx, Python, Django | PostgreSQL ("Shared DB -workflow/legitrack") |
| **Audittool PROD (Windows Servers)** | Production environment for Audittool, a legacy application developed in Classic ASP, hosted on IIS web server with MySQL. | Classic ASP, Microsoft IIS | MySQL (on Windows Server) |
| **Audittool Staging (Windows Servers)** | Staging environment for the Audittool, similar to the production setup with IIS and MySQL. | Classic ASP, Microsoft IIS | MySQL (on Windows Server) |
| **EA Cloud PROD (Windows Servers)** | Enterprise Architect (EA) Cloud production environment hosted on a Windows server, connecting to DB server . | EACloud Server | MySQL |
| **EA Cloud Staging (Windows Servers)** | Staging environment for Enterprise Architect (EA) Cloud, set up similarly to the production environment. | EACloud Server | MySQL |
| **Databases** | Various MySQL and PostgreSQL databases supporting different applications. | - | Shared MySQL DB , Legitrack PROD DB , EA PROD DB |
| **Key Notes:**   * **Audittool**: Only the Audittool application is built using **Classic ASP** and hosted on **Microsoft IIS** with a **MySQL database**. * **Other Applications**: All other applications (Safetyweb, Legitrack, Interface, Workflow) are developed using **Python** and **Django**, utilizing microservice architecture, and **each runs on separate servers** with Rancher for management. * **Database Servers**:   + **Shared MySQL DB Server** : Shared database for multiple applications requiring workflow-related data.   + **Legitrack PROD DB Server**: Dedicated MySQL database for the Legitrack production environment.   + **EA PROD DB Server**: Dedicated MySQL database for the Enterprise Architect production environment. | | | |

# 

# **5. PROPOSED NEW ARCHITECTURE**



## **5.1 DETAIL OVERVIEW**

| **Layer** | **Component** | **Purpose/Role** | **Type/Technology** |
| --- | --- | --- | --- |
| **Application Layer** | Safetyweb App | Manages safety incident reporting, analysis, and investigations. | Independent Service |
| Legitrack App | Manages regulatory tracking, creating new regulations, and compliance. | Independent Service |
| Workflow App | Manages workflows related to operational processes and tasks within an organization. | Independent Service |
| Audit tool App | Facilitates audit preparation, Corrective Action Plans (CAP), and compliance standards, especially in aviation contexts. | Independent Service |
| Enterprise Architect App | Assists in designing and maintaining processes and architectures to align with business goals. | Independent Service |
| **Backend Layer** | Safetyweb Backend | Backend service for handling requests, processing data, and interfacing with the Safetyweb database. | Node.js (MERN Stack) |
| Legitrack Backend | Backend service for managing regulations, compliance data, and interfacing with the Legitrack database. | Node.js (MERN Stack) |
| Workflow Backend | Backend service for handling workflows, task management, and integration with the Workflow database. | Node.js (MERN Stack) |
| Audittool Backend | Backend service for audit management, interfacing with the Audittool database, and performing scheduled jobs. | Node.js (MERN Stack) |
| Enterprise Architect Backend | Backend service for designing organizational processes, interfacing with the Enterprise Architect database, and ensuring alignment with business. | Node.js (MERN Stack) |
| **Database Layer** | PostgreSQL (Safetyweb) | Relational database for storing safety incident data, reports, and analysis results. | PostgreSQL |
| MYSQL (Legitrack) | Relational database for managing regulatory compliance data and annotations. | MYSQL |
| MYSQL(Workflow) | Relational database for storing workflow processes, tasks, and related operational data. | MYSQL |
| MySQL (Audittool) | Relational database for managing audit data, CAPs, and related documentation, historical reasons for MySQL choice. | MYSQL |
| MYSQL(Enterprise Architect) | Relational database for storing process designs, architectures, and related organizational models. | MYSQL |

## 

## **5.2 PROPOSED NEW TECHNOLOGY STACK**

The Table below provides a detailed overview of the recommended technology stack for design, development, and data communication across application layers.

| **Platform** | **Description/Use** |
| --- | --- |
| **Design and Prototyping** | |
| **Figma** | Cloud-based design tool for UI/UX design, interactive prototypes, and team collaboration. |
| **Frontend** | |
| **React** | Javascript library for building user interfaces with a component-based architecture. |
| **TypeScript** | Superset of JavaScript adding static typing to improve code quality and maintainability. |
| **Semantic HTML** | Uses meaningful HTML elements for accessibility and SEO. |
| **MUI (Material-UI)** | React UI framework providing pre-designed components based on Material Design principles. |
| **Tailwind CSS** | Utility-first CSS framework for creating custom designs directly in the markup. |
| **Backend** | |
| **Node.js** | JavaScript runtime for building fast, scalable network applications with a non-blocking architecture. |
| **TypeScript** | Enhances Node.js with static typing to improve code quality and maintainability. |
| **Database** | |
| **PostgreSQL** | Open-source relational database known for its extensibility, complex queries, and reliability. |
| **MySQL** | Widely used open-source relational database, preferred for its speed and simplicity. |
| **Cloud Hosting** | |
| **Cloud (AWS/Azure)** | Cloud platform providing scalable compute, storage, and database services. |
| OR | |
| **Microsoft Azure** | Cloud platform offering services for deploying, managing, and scaling applications with strong Microsoft integration. |
| **DevOps Tools** | |
| **Argo CI/CD** | Tool for continuous integration and delivery workflows on Kubernetes. |
| **Rancher** | Manages Kubernetes clusters, simplifying deployment and management. |
| **Docker** | Containerization tool for consistent application deployment. |
| **Kubernetes** | Orchestration tool for automating deployment, scaling, and management of containerized applications. |
| **Version Control** | |
| **Git** | Distributed version control system for tracking code changes. |
| **GitHub** | Platform for Git repository hosting with collaboration features. |
| **Project Management** | |
| **Jira** | Tool for agile project management, supporting scrum boards, kanban boards, and issue tracking. |
| **Trello** | Visual project management tool using boards, lists, and cards for task organization. |
| **API Documentation** | |
| **Swagger** | Tool for designing, building, and documenting RESTful APIs with interactive documentation. |
| **Testing Frameworks** |  |
| **Cypress** | End-to-end testing framework for web applications, providing reliable and easy-to-write tests. |
| **Swagger (API Testing)** | Used for testing RESTful APIs based on the API documentation. |
| **Key Notes:**   * **Frontend & Backend Synergy:** TypeScript's use in both frontend and backend provides consistency and reduces cognitive load for developers. * **Cloud Hosting:** AWS and Microsoft Azure offer scalable, reliable infrastructure with global reach**.** * **DevOps & CI/CD:** Tools like Argo CI/CD, Rancher, Docker, and Kubernetes support a robust DevOps pipeline for efficient microservices deployment. * **Testing & Quality Assurance:** Cypress and Swagger ensure high code quality with comprehensive testing for web applications and APIs. | |

## **5.3 TECH STACK COMPARISON: Rationale for Upgrading**

Below is the comparison analysis between current stack vs new proposed stack along with the scalability benefits :

| **Category** | **Existing Technology** | **Recommended Technology** | **Benefits of Recommended Technology** |
| --- | --- | --- | --- |
| **Design and Prototyping** | N/a | Figma | - Figma: Cloud-based, real-time collaboration, interactive prototypes, and high-quality UI/UX design capabilities. |
| **Frontend** | Python/Django with traditional HTML/CSS/JavaScript | React, TypeScript, Semantic HTML, MUI, Tailwind CSS | - React: Component-based architecture for dynamic, high-performance UIs.  - TypeScript: Static typing for improved code quality.  - Semantic HTML: Enhances accessibility and SEO.  - MUI: Pre-designed components for consistent design.  - Tailwind CSS: Utility-first CSS for custom, responsive designs. |
| **Backend** | Python, Django, Classic ASP | Node.js, TypeScript | - Node.js: Non-blocking, event-driven architecture ideal for scalable applications.  - TypeScript: Improves code maintainability with static typing.  - Express.js (commonly used with Node.js): Simplifies API development. |
| **Database** | PostgreSQL, MySQL | PostgreSQL, MySQL | - PostgreSQL: Advanced features for complex queries and high data integrity.  - MySQL: Performance and simplicity for specific use cases. |
| **Cloud Hosting** | AWS, Microsoft Azure | AWS/ Microsoft Azure | - AWS & Microsoft Azure: Reliable, scalable cloud platforms with a range of managed services. No change needed. |
| **DevOps Tools** | N/a | Argo CI/CD, Rancher, Jenkins, Docker, Kubernetes | - Argo CI/CD: Automates integration and delivery on Kubernetes.  - Rancher: Simplifies Kubernetes cluster management.  - Jenkins: Customizable CI/CD automation.  - Docker: Consistent deployment with containerization.  - Kubernetes: Efficient container orchestration for scaling and managing applications. |
| **Version Control** | Github | Github | - Git: Modern version control system supporting collaborative development.  - GitHub: Repository hosting with collaboration features and CI/CD integrations. |
| **Project Management** | Traditional project management tools or methods | Jira /Trello | - Jira: Comprehensive agile project management with support for sprints, kanban, and reporting.  - Trello: Visual task management with boards, lists, and cards. |
| **API Documentation** | Manual documentation or basic tools | Swagger |  |
| **Testing Frameworks** | Basic or manual testing methods | Cypress, Swagger (API Testing) |  |

# **6. INFRASTRUCTURE**

To design a comprehensive infrastructure for the complete application development, we need the following proposed infrastructure. However, some of them are optional and we may decide on the go.

| **Technology/Tool** | **Tech Overview** |
| --- | --- |
| **Docker** | * Containerization: All components of the application (backend, database, etc.) should be containerized using Docker to ensure consistency across environments. * Docker Compose: Use Docker Compose to manage multi-container Docker applications, defining services (e.g., backend, database, storage) and their interactions in a single file. * Docker Registry: A private or public Docker registry to store and manage Docker images. |
| **Server** | * Load Balancer: Use a load balancer (e.g., NGINX, HAProxy) to distribute incoming traffic across multiple servers. * Web Server: NGINX or Apache to serve the web application. Ensure SSL/TLS is configured for HTTPS. * Reverse Proxy: NGINX can act as a reverse proxy, handling incoming HTTP requests and forwarding them to the backend services. |

| **Database (PostgreSQL)** | * High Availability: Implement PostgreSQL in a high-availability setup using replication (e.g., streaming replication). * Backup and Recovery: Set up regular backups using tools like pg\_dump or managed solutions like pgBackRest, with automated scripts to ensure timely backups. * Database Security: Ensure SSL connections to the database, encrypt sensitive data, and manage access control through roles and permissions. |
| --- | --- |
| **Storage (NFS)** | * Network File System (NFS): Use NFS for shared storage across multiple servers. It can be used for static files, backups, or shared application data. Example : S3 bucket. * Performance Optimization: Configure NFS with caching mechanisms to enhance performance. Use NFS v4 for better security and performance. * Redundancy and Availability: Ensure NFS storage is highly available and redundant, pos |
| **CI/CD (Argo/Rancher)** | * GitOps Workflow: Implement a GitOps workflow where any change in the Git repository triggers a deployment pipeline. * Automated Testing: Integrate automated testing (unit, end-to-end) within the CI pipeline. * Deployment Strategies: Utilize strategies like Blue/Green deployments, and rolling updates to minimize downtime. |
| **Security** | * Infrastructure Security:   + Use firewalls (e.g. Azure NSG) to control inbound and outbound traffic.   + Harden server configurations by disabling unused services and ports.   + Use VPNs or private networking to limit access to sensitive components. * Application Security:   + Implement OAuth2 or JWT for authentication.   + Use HTTPS for secure communication.   + Regularly update dependencies and perform vulnerability scanning. * Database Security:   + Hashing data related to secure password.   + Use different roles and policies for access control. |
| **Backup** | * Database Backup:   + Automated daily backups with incremental backups throughout the day.   + Store backups on a separate, secure storage (e.g., S3, NFS).   + Regularly test backup restoration to ensure reliability. * Application Data Backup:   + Regular backups of application data stored in NFS or other storage solutions.   + Versioned backups to allow restoration of previous versions. * Disaster Recovery:   + Define a disaster recovery plan that includes RTO (Recovery Time Objective) and RPO (Recovery Point Objective). |
| **Scaling** | * Horizontal Scaling:   + Use Kubernetes or Docker Swarm to manage containers, allowing for automatic scaling based on load. * Vertical Scaling:   + Configure auto-scaling for server resources (CPU, RAM) based on demand. |
| **Networking** | * DNS Management: Use a managed DNS service (e.g., Route 53, Cloudflare) for handling domain resolution and traffic routing. * Content Delivery Network (CDN): Use a CDN to cache and distribute static content globally, reducing load times. |
| **Development Environment** | * Local Development: Ensure a consistent development environment using Docker or Vagrant, mirroring production as closely as possible. * Staging Environment: A staging environment identical to production for final testing before deployment. |

## 

# **7. REQUIRED SYSTEM CONFIG**

To provide a comprehensive overview of the required CPU, RAM, and disk space for a system that includes Docker, and PostgreSQL, here’s a breakdown based on typical usage scenario

| **Component** | **CPU** | **RAM** | **Disk** | **Notes** |
| --- | --- | --- | --- | --- |
| **Docker** | Minimum: 2 vCPUs | Minimum: 4 GB | Minimum: 20 GB | Dependent on the number and type of containers |
| Recommended: 4-8 vCPUs | Recommended: 8-16 GB | Recommended: 100 GB+ | Higher disk space for persistent volumes and multiple images |
| High Workload: 8+ vCPUs | High Workload: 16+ GB | High Workload: SSD recommended |  |
| **PostgreSQL** | Minimum: 2 vCPUs | Minimum: 4 GB | Minimum: 100 GB | Dependent on database size and query complexity |
| Recommended: 4-8 vCPUs | Recommended: 8-16 GB | Recommended: 250 GB - 1 TB | Consider RAID for redundancy and performance |
| High Workload: 8+ vCPUs | High Workload: 32+ GB | High Workload: Multiple TBs with SSDs | Particularly important for large datasets and high concurrency |
| **Node** | Minimum: 1-2 vCPUs | Minimum: 1-2 GB | Minimum: 10 GB | Dependent on application complexity and traffic |
| Recommended: 4 vCPUs | Recommended: 4-8 GB | Recommended: 20-50 GB | Higher specs needed for CPU-intensive or large-scale applications |
|  | High Traffic: 8+ vCPUs | High Traffic: 16+ GB | High Traffic: SSD recommended | Consider scaling horizontally with more instances |

## 

# **8. TECHNICAL SPECIFICATION - SCOPE OF WORK**

Technical Specifications outline the project's functional requirements and provide a comprehensive breakdown of each microservice and it’s module, ensuring clear guidance throughout the development process and planning stages.

## **8.1 Safety Web - Microservice**

| **Module** | **Module Description** |
| --- | --- |
| **Incident / Occurrence Reporting** | |
| **All** | All Incident reports view |
| **Initial Assessment** | In this view either mark occurrences as accepted or define actions that are required immediately. Both actions change the occurrence status from initial to open. |
| **Review** | Incident data is review and listing report |
| **Close multiple** | Access to incident reports closing |
| **Lock multiple** | TDB |
| **Investigation** | Implement as per current view |
| **Locked** | Overview of Occurrence/incident reports |
| **Other Modules** | |
| **Analysis** | Duplicates , logs , probabilities, Related reports |
| **Activities** | Your activities and Organization activities |
| **Message** | Inbox |
| **Auth Module** | |
| **Auth** | Handles authentication processes such as login, logout, password reset, etc. |

| **Taxonomy** | | |
| --- | --- | --- |
| **Module** | **Feature** | **Description** |
| **Occurrence Reporting** | Occurrence Categories | Define and categorize different types of aviation occurrences (accidents, incidents, etc.). |
|  | Event Types | Classify events leading to an occurrence, such as loss of control or runway excursion. |
|  | Phase of Flight | Identify and record the phase of flight during which the event occurred. |
| **Contributing Factors** | Human Factors | Record and analyze human errors or decisions that contributed to the occurrence. |
|  | Environmental Factors | Capture environmental conditions (e.g., weather) that contributed to the occurrence. |
|  | Mechanical Issues | Log any mechanical or technical failures contributing to the occurrence. |
|  | Organizational Factors | Analyze organizational or procedural shortcomings that led to the occurrence. |
| **Severity Classification** | Severity Levels | Classify the severity of the occurrence (minor, serious, catastrophic, etc.). |
| **Injury and Damage Reporting** | Injury Reporting | Record and categorize injuries to passengers, crew, and third parties. |
|  | Damage Assessment | Log the extent of damage to the aircraft and other properties. |
| **Safety Recommendations** | Recommendation Generation | Generate safety recommendations based on the investigation findings. |
| **Narrative Descriptions** | Detailed Event Narratives | Provide detailed descriptions of the occurrence, including the sequence of events. |
| **Data Standardization** | ADREP Compliance | Ensure all data is compliant with the ICAO ADREP standards. |
| **Integration and Reporting** | Data Integration | Integrate ADREP data with existing systems for comprehensive reporting and analysis. |
|  | Custom Reporting | Develop custom reports based on ADREP data for various stakeholders. |
| **API Root** | Base endpoint for all ADREP-related functionalities. | |
| **Event Types API** | Manage event type classifications via API. | |
| **Phase of Flight API** | Handle phase of flight data through API calls. | |
| **Contributing Factors API** | Record and retrieve contributing factors through API. | |
| **Admin Interface** | Administrative interface for managing ADREP taxonomy and data. | |
| **Incident Reporting View** | User-friendly interface for recording and managing incidents based on ADREP taxonomy. | |
| **Analysis Dashboard** | Dashboard for analyzing incidents using ADREP taxonomy data. | |

## **8.2 Legitrack - Microservice**

| **Module** | **Functions** |
| --- | --- |
| **Regulation Builder** | * Index * Create * Images * Images upload * Edit * Checkout * Checking * Unpublish * Delete request * Inform check in * Delete * Archive * Unarchive * View * File * Pdf export |
| **Regulation Export** | * Index * Preview |
| **Requirements** | * List * Export * Create * Import * Update * Delete |
| **Tools** | * List |
| **Timetrack** | * Index * Create * View * Edit * Delete |
| **Support** | * Index |
| **Missing Scope** | * Since we do not have login access to the platform. So, the provided outline is based on shared documents, video discussions, and high level code review analysis. This table presents a comprehensive view of the project’s tasks and modules. Please ensure to incorporate any missing scope identified during project initiation as needed. |

## **8.3 AuditTool - Microservice**

| **Module** |
| --- |
| * About us |
| * Access Denied |
| * Action |
| * Client |
| * Finding |
| * Finding session |
| * Item |
| * User |
| * Audit result |
| * Audits |
| * Captcha |
| * Client Data history |
| * Clients |
| * Compliance Matrix |
| * Contact |
| * Cost Estimate for update |
| * Maintenance |
| * Default |
| * Doa form |
| * Docs |
| * EAS audit |
| * EAS regulations |
| * Elevator pitch |
| * Forgot password |
| * Finding session access list |
| * SARP |
| * Internal ADSE DOA |
| * Login |
| * Logout |
| * Logs |
| * Mod client |
| * Mod item |
| * Mod user |
| * News |
| * Prepare Audit |
| * Print |
| * Priority |
| * Processed docs |
| * Progress CE |
| * Progress |
| * Push documents |
| * Release action |
| * Reset Password |
| * Result Assignments |
| * Result Detail |
| * RWS form |
| * Source documents |
| * Status |
| * Support |
| * Unauthorize |
| * upload ccs |
| * upload pqs |
| * web items |
| * Wizard |
| * Work group assignments |
| * **Missing Scope :** Since we do not have login access to the platform. So, the provided outline is based on shared documents, video discussions, and high level code review analysis. This table presents a comprehensive view of the project’s tasks and modules. Please ensure to incorporate any missing scope identified during project initiation as needed. |

## **8.4 Enterprise Architect - Microservice**

| **Features** | **Task** |
| --- | --- |
| **Missing Scope** | * Since we do not have login access to the platform. So, the provided outline is based on shared documents, video discussions, and high level code review analysis. This table presents a comprehensive view of the project’s tasks and modules. Please ensure to incorporate any missing scope identified during project initiation as needed. |

## **8.5 Workflow - Microservice**

| **Features** | **Task** |
| --- | --- |
| **Medical** | * Apply |
| **Accounts** | * Login * Logout * Register |
| **Core** | * Dashboard * Application list * Application details |
| **Missing Scope** | * Since we do not have login access to the platform. So, the provided outline is based on shared documents, video discussions, and high level code review analysis. This table presents a comprehensive view of the project’s tasks and modules. Please ensure to incorporate any missing scope identified during project initiation as needed. |

## **8.6 Investigation - Microservice**

| **Feature** | **Description** |
| --- | --- |
| **Investigation** | * List * Create * Detail * Eccairs Detail * Update * Delete |
|
|
|
|
|
| **Team** | * Create * Update * Delete * Detail |
|
|
|
| **Team Member** | * Create * Update * Delete |
|
|
| **State Roles** | * Determine * Create * Update * Delete |
|
|
|
| **Investigation Contacts** | * Detail * Create * Delete * Create State * Create IIC * Create IC * Create Adviser * Create Participant * Create AccRep * Create Observer * Create Job * Create Group * Create Coordinator |
|
|
|
|
|
|
|
|
|
|
|
|
| **Actions** | * Toggle Complete |
| **Logs** | * Investigation Logs |
| **Notes** | * Investigation Notes * Investigation Notes Create * Update * Generate PDF * Note Format |
|
|
|
|
| **Collaboration App** | * Task * Task Create |
|
| **Documents** | * Preview * Preview Contact |
|
| **Decision-Support** | * Index * Investigate |
|
| **Importer** | * Webhook > New Investigation |
| **Notifications** | * Form * Edit * Download * Inventory |
|
|
|
| **File** | * Upload * Investigation Files * Folder Files |
|
|
| **Colab** | * Task List * Task Details * Task Create * Task Update * Task Delete * Task PDF Generate |
|
|
|
|
|
| **Documents** | * Detail * Letter List * Letter Create * Letter Update * Letter Delete |
|
|
|
|
| **Missing Scope** | **Note:** Since we do not have access to the actual modules, the provided outline is based on shared documents, video discussions, and system analysis. This table presents a comprehensive view of the project’s tasks and modules. Please ensure to incorporate any missing scope identified during project initiation as needed |

## 

# **9. TEAM STRUCTURE**

| **Team** | **Role** |
| --- | --- |
| **Project Manager** | Manages the entire project from scoping to delivery |
| **Application Architect** | Designs the Application architecture |
| **UI/UX Designer** | Designs the UI of the application |
| **Frontend Developers** | Develop the Frontend of the web app from scratch and consumes API to make it fully functional |
| **Backend Developers** | Develop the APIs and backend of the web app from scratch |
| **QA Testing** | Writes test cases and tests the application |

# **10. APPLICATION TRANSITION AND IMPLEMENTATION PLAN**

This plan provides a detailed approach to managing the application migration and technology stack transition for QCAA, ensuring smooth implementation and alignment with the organization's goals.

| **Phase** | **Tasks** | **Responsible Party** |
| --- | --- | --- |
| **Phase 1: Planning** | • Define migration objectives and scope. | Developer, QCAA |
| • Conduct a detailed assessment of existing infrastructure and applications. | Developer |
| • Develop a comprehensive migration strategy and risk assessment, including roll-back plans. | Developer |
| • Identify key stakeholders and assign roles and responsibilities. | QCAA |
| • Plan budget, resources, and timelines for AWS, Docker, Kubernetes, and other services acquisition. | QCAA |
| **Phase 2: Infrastructure Setup** | • Set up an AWS cloud environment, including EC2, RDS, S3, VPC, IAM, and networking configurations. | Developer |
| • Configure CI/CD pipelines using Argo, and GitHub. | Developer |
| • Install and configure Docker and Kubernetes (EKS) for container management. | Developer |
| • Implement cloud monitoring, security, and compliance configurations (CloudWatch, CloudTrail, WAF, Shield). | Developer |
| **Phase 3: Application Migration** | • Migrate existing applications (Safetyweb, Legitrack, Interface, Workflow) to AWS using containerized Docker environments. | Developer |
| • Redevelop the Audittool using the MERN stack and deploy it on AWS. | Developer |
| • Implement data migration strategies for PostgreSQL and MySQL databases. | Developer, QCAA |
| • Ensure all applications are containerized and deployed using Kubernetes on AWS EKS. | Developer |
| • Perform initial testing and validation in the staging environment. | Developer |
| **Phase 4: Testing and Validation** | • Conduct comprehensive testing (functional, integration, performance, security) of all migrated and redeveloped applications in the new environment. | Developer, QCAA |
| • Utilize Swagger and Cypress for API and end-to-end testing to ensure application functionality and integrity. | Developer |
| • Perform user acceptance testing (UAT) with QCAA stakeholders. | Developer, QCAA |
| • Address any issues or bugs identified during testing. | Developer |
| **Phase 5: Training and Documentation** | • Conduct training sessions for QCAA technical teams and end-users on new systems, AWS, Docker, and Kubernetes management. | Developer |
| • Develop and provide documentation on new processes, backup/recovery, security protocols, and support. | Developer |
| • Share API documentation and usage guidelines using Swagger. | Developer |
| **Phase 6: Go-Live** | • Perform final go-live checklist and system readiness assessment. | Developer, QCAA |
| • Deploy all applications to production AWS environment. | Developer |
| • Monitor initial performance closely and ensure support teams are available for any issues. | Developer, QCAA |
| **Phase 7: Post-Go-Live Support** | • Provide continuous support and monitoring for the first month post-migration to address any unforeseen issues and optimize performance. | 75way |
| • Review and optimize AWS costs, cloud resource usage, and scaling policies. | 75way |
| • Conduct post-migration review and gather feedback from QCAA stakeholders for future improvements. | Developer QCAA |
| **Key Considerations:**   1. **Communication Plan**: Regular status meetings, reports, and checkpoints with all stakeholders to ensure transparency and alignment. 2. **Risk Management**: Develop contingency plans and establish a rollback strategy to handle potential challenges. 3. **Security and Compliance**: Ensure that all processes adhere to QCAA's security policies and regulatory requirements throughout the transition. 4. **Resource Management**: Monitor resource allocation for cost-effectiveness and operational efficiency. | | |

# **11. TIME & COST ESTIMATION**

| **Estimates** | **Time (in months)** | **Cost (in QAR)** |
| --- | --- | --- |
| **Total** | **6-8** | **100K - 150K QAR** |

# **12. MAINTENANCE & SUPPORT TERMS**

| **Category** | **Details** | **Cost** |
| --- | --- | --- |
| 1. Maintenance Services | | |
| **Scope** | Regular updates, bug fixes, and performance improvements. | $[Placeholder] per month. |
| **Frequency** | Monthly updates and quarterly reviews. |  |
| 2. Support Services | | |
| **Support Levels** | Basic Support: Email and phone support during business hours.  Advanced Support: Includes critical issue resolution, available 24/7. |  |
| **Basic Support Cost** |  | $[Placeholder] per month. |
| **Advanced Support Cost** |  | $[Placeholder] per month. |
| **3. Service Level Agreement (SLA)** | | |
| **Uptime Guarantee** | [Placeholder]% uptime. |  |
| **Response Time** | Initial response within [Placeholder] hours. |  |
| **4. Term and Renewal** | | |
| **Initial Term** | [Placeholder] months/years. |  |
| **Renewal** | Automatic renewal unless terminated with [Placeholder] days' notice. |  |
| **Termination** | Notice required [Placeholder] days in advance. |  |
| **5. Additional Services** | | |
| **Custom Requests** |  | $[Placeholder] per hour. |
| **On-Demand Support** |  | $[Placeholder] per hour. |
| **6. Payment Terms** | | |
| **Payment Schedule** |  | Monthly/Annual payments. |

In case of any query, please feel free to ask.

**Thanks & Regards  
Team 75way   
lokesh@75way.com  
Skype ID: lbansal2099   
Whatsapp : +91-8802204052   
https://75way.com**