RE Workflow Solutions
Preliminary System Design, Security & Technical Document

Preliminary Approach & Solution Analysis Document

25-August-2025
Version 1.0

Contents

1 Objective

Automate the Dealer Development process to streamline prospect applications, vacancy checks, approvals, and LOI issuance. Enhance efficiency and scalability for 1,000 users, ensuring a seamless dealer onboarding experience. Maintain DPDP compliance with secure, reliable processes, reducing manual effort and improving decision-making for stakeholders.

Additionally, the review and approval process differs from scenario to scenario, so the system should be dynamically capable enough to cater this need.

2 Problem Statement

The absence of a Dealer Development automation system creates significant challenges in managing the dealer onboarding process efficiently and securely. Currently, manual processes for handling prospect applications, vacancy checks, approvals, and LOI issuance are time-consuming, error-prone, and lack scalability.

This leads to delays in processing applications, inconsistent data handling, and difficulties in generating timely reports, such as daily web lead summaries and location-wise details. Additionally, the lack of automated security measures increases the security risk of non-compliance with Digital Personal Data Protection (DPDP) regulations, particularly for sensitive prospect data. Without a centralized system, stakeholder coordination (DD Lead, RBM, ZBH, NBH) is fragmented, hindering decision-making and impacting the overall dealer onboarding experience.

3 System Requirements

3.1 Functional Requirements

1. **Dealer Onboarding Workflow**

    - Capture enquiries via form, validate against opportunity list.
    - Automated emails, reminders, scoring, and top-candidate shortlisting.
    - Multi-level approvals (ASM → ZM → RBM → ZBH → NBH).
    - Profile sheet submission, KT evaluations, and final LOI issuance.
    - Custom Approval Workflows: Admin can configure different approval chains depending on region size (e.g., simpler workflows for smaller regions, extended hierarchies for larger regions).

2. **Dealer Exit Workflow**

- **Resignation:** Letter submission, ASM template, approvals (ZBH → DD Lead → NBH), legal concurrence, dealer acknowledgment.
- **Termination:** Record reasons, prepare docs, legal show-cause notice (15-day timer), approvals (ZBH → DD Lead → NBH → CCO → CEO), final termination letter.
- **Full & Final Settlement:** Coordinate with stakeholders, reconcile dues, block SAP/MSD codes, Finance AR calculates F&F, dealer acceptance or legal escalation.

3. **Automation**

- Automated notifications (acknowledgments, reminders, approvals, LOI, F&F).
- SLA-driven reminders (Day 2, Day 5, 15-day cutoffs).
- Reminders can be customized in terms of duration frequency vs levels.

4. **Reporting**

- Every user can login and view the Dashboard & quick reports

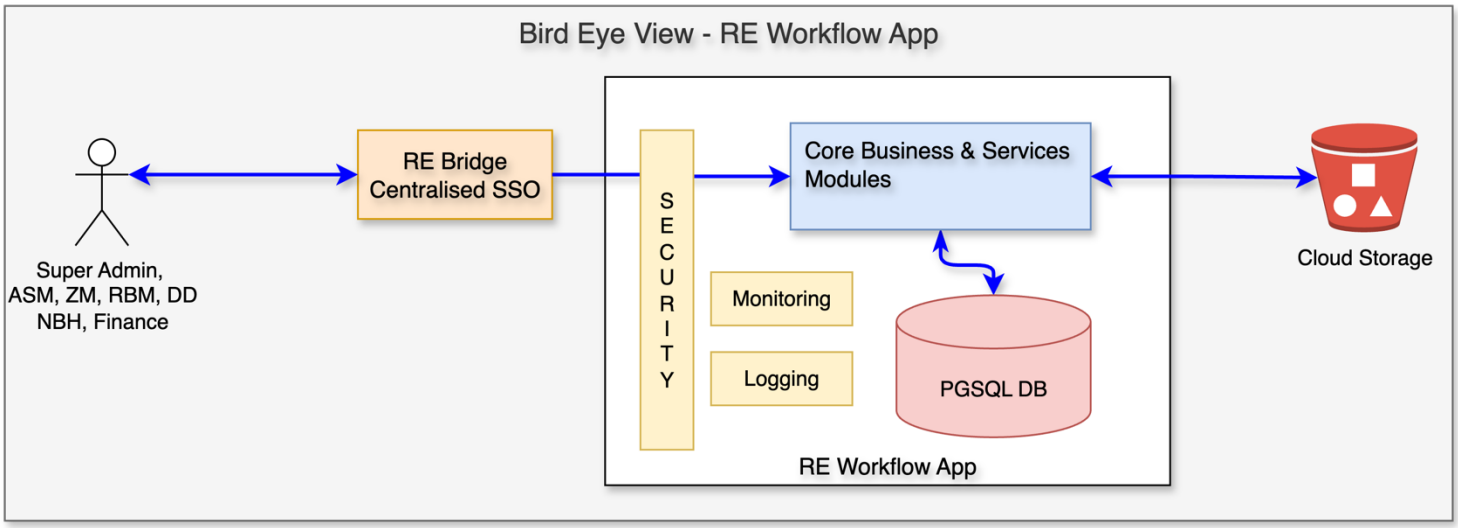
5. **Access Control**

- Role-based access for Admin, ASM, ZM, RBM, ZBH, NBH, CCO.
- Regional scoping and complete audit logs.

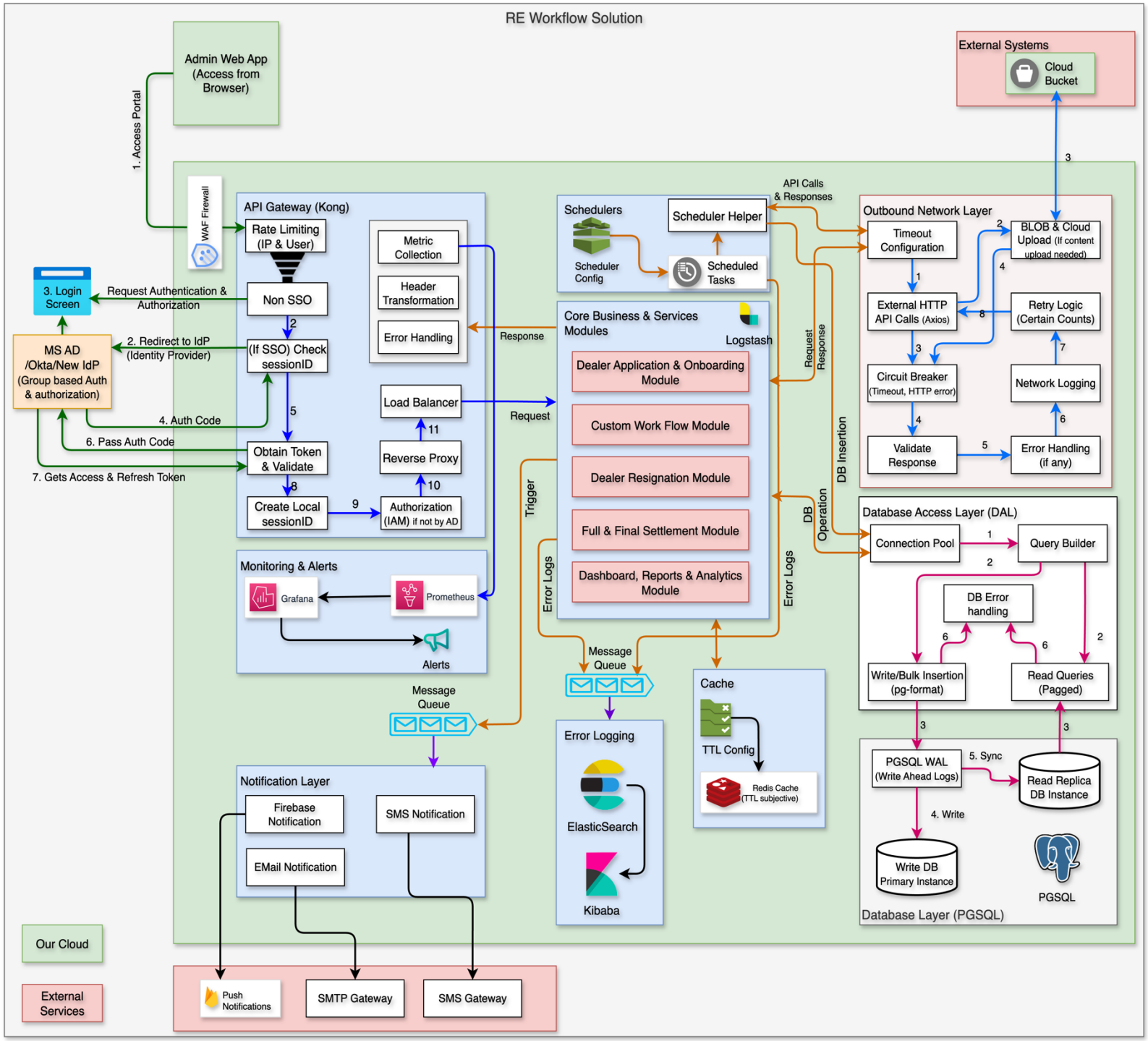
3.2 Non-Functional Requirements

1. **Scalability**: Support 10,000 total users and 200 concurrent users.
2. **Performance**: Achieve <2s latency for most operations during peak hours (9 AM–5 PM IST).
3. **Availability**: Ensure 99.9% uptime.
4. **Security**: Comply with DPDP Act, 2023, with encryption and RBAC.
5. **Reliability**: Maintain data consistency and fault tolerance across workflows.

4 Bird Eye View



5 System Design



6 System Design Explanation

6.1 User Authentication & Access Flow

The system leverages **Royal Enfield's existing Active Directory (AD)** via an **Identity Provider (IdP)** (Azure AD / ADFS) for authentication. This means that user identity and credentials are not managed within the portal but by AD, ensuring consistency with corporate policies.

6.1.1 Step-by-Step Authentication Flow

First-Time Access

1. A user navigates to the portal (e.g., https://dd-portal.re.com/).
2. The portal checks for an active application session (secure session cookie sid). If absent/expired, the user is redirected to the IdP's **/authorize** endpoint.
3. The IdP validates the user credentials against AD
4. Upon successful authentication, the IdP (Identity Provider) redirects back to the portal's callback URL with an authorization code.
5. The portal backend exchanges the authorization code with the IdP's **/token** endpoint over HTTPS. Response includes: **ID Token (identity claims), Access Token, and Refresh Token**.
6. The ID Token is validated using IdP's public keys and claims are extracted (user, roles, groups).
7. A secure **application session** is created (stored locally), and an opaque sid cookie is set in the user's browser.
8. User gains access to the portal and their dashboard.

Subsequent Access

1. On the next request, the browser presents the sid cookie.
2. If the session is still valid, the request is allowed directly.
3. If expired, the portal redirects to the IdP.

    - If the IdP SSO session is still valid, a new token is issued silently via Refresh Token (no login prompt).
    - If IdP SSO is also expired, the user must re-enter credentials.

6.2 API Driven Framework

Expose all business capabilities (Applications, Evaluations, Approvals, Documents, Notifications, F&F, Reports) as **versioned, secure APIs**. This decouples frontends (web portal today, mobile app tomorrow) from backend services and enables internal/partner integrations without re-engineering.

6.3 Frontend Layer (React.js)

The frontend of the Dealer Development Automation system will be a **web portal** accessible to applicants and internal staff. It will provide intuitive **forms for dealer applications**, interactive **dashboards for approvals and evaluations**, and comprehensive **reporting views** (daily summaries, location-wise performance, F&F status). The portal will communicate exclusively with backend services through the **API Gateway**, ensuring a clean separation of concerns and enabling an API-driven architecture.
The frontend will be developed using **ReactJS**, chosen for its **strong ecosystem and wide community support available**. React allows us to build highly responsive UIs where approval workflows, reminders, and multi-level evaluation forms can be represented as reusable components, reducing development effort and improving maintainability.

6.4 Backend Service (Node.js)

Node.js has become one of the most widely adopted backend technologies because of its API First, **event-driven, non-blocking I/O architecture**, which makes it highly efficient at handling thousands of concurrent API calls, webhooks, and real-time events with minimal resource usage. The combination of performance, developer productivity, ecosystem maturity, and platform support makes Node.js not just a trending choice, but a **future-proof technology** for building scalable, API-driven enterprise systems like the Dealer Development Automation platform

6.5 Data Handling & Storage

1. **PostgreSQL** : Database: Primary transactional store; read replica for reporting.
2. **Redis Cache**: Session storage, rate limiting, reminder state.
3. **Object Storage**: Legal and operational document archive over cloud storage
4. **Audit Logs**: Immutable records of all approvals, actions, and data changes for compliance.
5. **Monitoring & Logging**: Prometheus/Grafana for metrics, ELK/Cloud Logging for logs.

7 Scalability & Performance

1. **Load Handling**: 1,000 requests/minute (200 users × 5 calls/minute).

2. **API Gateway**: Handles rate limiting, load balancing, routing.
3. **Database**:
    1. PostgreSQL: Read replicas for queries.
    2. Object Storage: Scalable for documents.
4. **Caching**: Redis for frequent queries (e.g., user roles, town lists).
5. **Asynchronous Processing**: Message Queue for emails and integrations.
6. **Microservice-ready design** allows scaling each service independently.
7. **Event-driven architecture** ensures async notifications and workflows.
8. **Configurable workflows** support regional customization.
9. **Mobile-ready design** with JWT support for future app extensions.
10. **Libraries & Dependencies** We ensure to use the libraries which are having healthy cadence & footprint, check known issues on it's GIT, Licence, ran by foundation or single person & GIT Insights

8 DPDP Act Compliance

The system adheres to the **Digital Personal Data Protection Act, 2023**, with the following measures:

1. **Data Minimization**: Collect only necessary data (e.g., dealer name, contact, financial details).
2. **Consent Management**: Obtain explicit consent via web forms, stored in a DB
3. **Data Subject Rights**: Provide interface for dealers to access their data.
4. **Data Security**: Use TLS 1.3 for data in transit and AES-256 for data at rest.
5. **Data Localization**: Deploy databases and storage in India-based cloud regions.
6. **Breach Notification**: Alerts within 72 hours of a data breach.
7. **Audit Trails**: Log all data access and modifications for compliance audits.

9 Security Requirements and Technical Implementation

1. **DPDP-Specific**:
    1. **Data Localization**: Deploy in India-based regions.
    2. **Consent Management**: Store consent records, provide opt-out.
    3. **Breach Notification**: Alerts within 72 hours
    4. **Audit Trails**: Log data access in ELK Stack.
2. **General Security**:
    1. **Encryption**: TLS 1.3, AES-256.
    2. **WAF**: Deploy cloud specific WAF.
    3. **MFA**: Enforce for sensitive roles.
    4. **Secure APIs**: Use API keys, OAuth tokens, rate limiting.
    5. **API Gateway / Reverse Proxy:** TLS termination, rate limiting, request validation, and routing.

6. **Network Firewall:** Ensures only HTTPS (443) is exposed externally; internal DBs remain private.
7. **Secrets Management:** All API keys, DB credentials, and OAuth secrets stored in KMS/Vault.

9.1 Input Validation and Sanitization

1. **Requirement**: Mitigate injection attacks (e.g., SQL injection, Cross-Site Scripting [XSS]) on user inputs such as questionnaire responses, comments, and custom flow configurations.
2. **Rationale**: Malicious inputs could execute unauthorized scripts or corrupt data, particularly in the platform's interactive forms and dynamic workflows.
3. **Risk Level**: Medium
4. **Implementation Strategy**:
    1. **Validation**: Enforce input schemas in Node.js, validating data types (e.g., email regex, numeric scores) before processing.

9.2 Secure File Handling

1. **Requirement**: Safeguard uploaded files (e.g., business plans, financial documents, legal papers) against unauthorized access, tampering, or malware.
2. **Rationale**: Exposure of confidential business data could disrupt the onboarding process or lead to legal liabilities, given the platform's document-centric nature.
3. **Risk Level**: High
4. **Implementation Strategy**:
    1. **Storage**: Utilize a secure object storage service (e.g., AWS S3 with SSE-KMS) with private access policies and server-side encryption.
    2. **Access**: Generate time-bound pre-signed URLs (e.g., 300-second expiration) via Node.js routes, authenticated with user tokens.

9.3 DDoS and API Rate Limiting

1. **Requirement**: Defend against Distributed Denial of Service (DDoS) attacks and API abuse during peak onboarding activity.
2. **Rationale**: High-traffic features (e.g., report downloads, progress sharing) are vulnerable to overload, potentially disrupting service availability.

3. **Risk Level**: Medium
4. **Implementation Strategy**:
    1. **Rate Limiting**: Configure express-rate-limit in Node.js to cap API requests at 100 per minute per IP, with whitelisting for trusted sources.
    2. **DDoS Mitigation**: Integrate a CDN with Web Application Firewall (WAF) capabilities (e.g., Cloudflare) to filter malicious traffic and distribute load.

10 Monitoring & Maintenance

1. **Monitoring**: Prometheus for metrics, Grafana for dashboards.
2. **Logging**: ELK Stack for DPDP-compliant audits.
3. **Maintenance**: CI/CD pipeline with Jenkins/GitHub Actions.

11 Camunda Support

Camunda provides APIs to let external applications start, advance, and complete workflow steps. In **Camunda 7**, this is done via the **REST API** (e.g., `POST /process-definition/key/dealer_application/start` to start a new dealer application flow). In **Camunda 8 (Zeebe engine)**, APIs are available as **gRPC client libraries** (Java, Node.js, Go) or via a **REST gateway**; for example, a Node.js service can call `zbc.createProcessInstance({ bpmnProcessId: "dealer_application_v1", variables: { applicantId: "APP123" } })` whenever a dealer submits an application form. This integration allows the process instance to flow through approval tasks (ASM → ZM → NBH), with each stage visible in **Camunda Operate**, and bottlenecks/heatmaps available in **Camunda Optimize**. Camunda offers a free **Community Edition** (open source) for self-managed setups, and a **SaaS version (Camunda 8)** with usage-based pricing; as of 2025, SaaS plans start with a **free tier (up to 5 users and limited usage)**, then scale to **Team/Enterprise plans**based on process instance volume, storage, and enterprise features like Optimize, Tasklist, and SLA support.