Whitepaper: Arinda Multi-Tenant Loyalty Platform **2026-04-01**

1. Executive Summary

This solution is a multi-tenant loyalty platform designed to support customer-facing membership experiences and operational administration from a unified codebase. It combines a branded mobile app, an administrative web portal, and two backend APIs that separate member operations from management operations.

The platform's core value is operational consistency across multiple customer tenants while preserving customer-specific branding, access control boundaries, and configurable integrations. It enables customer administrators to manage members, loyalty levels, notices, QR visit types, and integration settings, while end users access loyalty status, perks, notices, bookings, and QR-based visit registration in the mobile app.

The implementation emphasizes secure authentication, tenant isolation, auditable changes, and repeatable CI/CD deployment to Azure and mobile distribution channels.

2. System Overview

The platform consists of:

- Mobile application layer:
    1. Flutter application with multi-brand support.
    2. Entra OTP login and backend session token usage.
    3. Loyalty, notices, bookings, profile, and QR scan flows.
- Web administration layer:
    1. React + TypeScript single-page application.
    2. MSAL-based authentication with Entra External ID.
    3. Customer management, user lifecycle, loyalty level and perk administration, notices, integrations, and QR visit administration.
- Backend services layer:
    1. Mobile API with customer-scoped endpoints and session token middleware.
    2. Management API with Entra token validation and customer authorization filters.
    3. Shared core library with entities, DbContext, helpers, middleware, integration contracts, and auditing primitives.
- Data and operations layer:
    1. SQL Server persistence via EF Core.
    2. GitHub Actions CI/CD workflows.
    3. Azure deployment targets for APIs and web portal.

3. Architecture

3.1 Logical Architecture

- Presentation tier:
    1. Flutter mobile client.
    2. React web admin client.
- Service tier:
    1. Mobile API for member operations.
    2. Management API for administrative operations.
- Domain and shared core tier:
    1. Entity definitions and domain relationships.
    2. Tenant context middleware.
    3. Token and session services.
    4. Integration hook contracts and dispatching.
- Data tier:
    1. SQL Server with EF Core mappings, indexes, and query filters.

3.2 Physical and Deployment Architecture

- Web portal is built and deployed to Azure Static Web Apps.
- Mobile API and Management API are built and deployed as separate Azure App Services.
- SQL Server is used as the persistence backend.
- Mobile binaries are produced in CI for Android and iOS, with workflows for Play Store and Apple Store/TestFlight delivery.

3.3 Technology Stack

- Backend:
    1. .NET 10.
    2. ASP.NET Core Web API.
    3. Entity Framework Core.
    4. SQL Server.
    5. Serilog console logging.
    6. OpenAPI/Scalar in development.
- Web:
    1. React.
    2. TypeScript.

3. Vite.
4. Zustand.
5. Axios.
6. MSAL Browser and MSAL React.

- Mobile:
  1. Flutter and Dart.
  2. Native OTP flow with Entra.
  3. Secure token storage.
  4. Customer-specific theming and assets.
  5. QR scanning.

3.4 Module Breakdown

- Backend core modules:
  1. Data model and DbContext.
  2. Customer context middleware.
  3. Session token middleware.
  4. Token exchange, refresh, and revocation services.
  5. Integration hook dispatch.
  6. QR payload signing and verification utilities.
- Mobile API modules:
  1. Auth controller.
  2. Loyalty and perk usage controller.
  3. Bookings controller.
  4. Notices controller.
- Management API modules:
  1. Customers and customer-user assignment controller.
  2. Users controller including signup completion and membership operations.
  3. Loyalty levels and perks controller.
  4. Notices controller.
  5. Integrations controller.
  6. QR visit types and logs controller.
  7. Profile controller.

- Web modules:
    1. Authentication service and token injection interceptor.
    2. Customer selection store and bootstrapping.
    3. Feature services aligned to management APIs.
    4. Role-aware dashboard layout.
- Mobile modules:
    1. Auth, API, loyalty, booking, notice, and profile services.
    2. Brand controller with customer asset overlays.
    3. Login and tabbed feature screens.
    4. QR parse and registration service.

3.5 Data Flow Diagrams in Text

- Mobile authentication and session establishment:
    1. User starts OTP sign-in against Entra using Entra Native Authentication.
    2. Mobile app receives Entra token.
    3. Mobile app calls mobile backend exchange endpoint with customer context.
    4. Backend validates Entra token and creates or restores user/customer linkage.
    5. Backend returns access and refresh tokens.
    6. Mobile app stores tokens securely and uses backend access token for API calls.
    7. On expiry, mobile app uses refresh endpoint for rotation.
- Web management request flow:
    1. User signs in through MSAL.
    2. Web app acquires Entra access token.
    3. Axios interceptor injects bearer token into Management API calls.
    4. Management API validates token, resolves user identity, applies customer authorization.
    5. API returns customer-scoped or admin-scoped data.
- QR visit registration flow:
    1. Admin creates QR visit type and print payload in management portal.
    2. QR payload includes customer id, type key, code id, and HMAC signature.
    3. Mobile scans and validates payload customer match.
    4. Mobile sends payload to registration endpoint with idempotency key.

5. Backend verifies signature and business constraints, logs visit event.

4. Mobile Application

4.1 Key Features

- Email OTP login via Entra native authentication flow with profile completion for signup path.
- Customer-scoped loyalty status, levels, perks, and usage history.
- Notices retrieval with audience targeting.
- Booking retrieval through integration-backed provider flow.
- QR scan and visit registration with cooldown/idempotency handling.
- Profile access and logout.
- Brand-specific UI, fonts, localization overlays, app naming, and flavors.

4.2 Mobile Architecture

- Service-oriented client architecture with shared API service wrapper.
- Central authentication service handling token lifecycle and authorization header generation.
- Strong separation between DTOs, services, and screen widgets.
- Runtime brand configuration loaded from customer asset hierarchy.
- Feature visibility controls based on customer access state.

4.3 Offline and Online Behavior

- The implementation is primarily online-first.
- A short-lived in-memory cache is implemented for next bookings retrieval.
- Mobile dependencies include local data libraries, but observed service flows call backend APIs directly for core operations.
- Session continuity is handled through secure refresh-token rotation rather than full local domain replication.

4.4 Security and Data Handling

- Access and refresh tokens are stored in secure mobile storage.
- API calls require authorization headers generated from current token state.
- Optional certificate pinning controls exist in configuration.
- QR payloads are verified with signed protocol payloads and customer match checks.

5. Web Admin Portal

5.1 Purpose and Capabilities

The portal is used by customer administrators and global administrators to manage customers, users, memberships, loyalty configuration, notices, integrations, and QR visit operations.

5.2 Architecture and UI Structure

- React routing with authenticated shell layout.
- MSAL provider initialization before app render.
- Zustand stores for authentication and customer selection.
- Service-layer abstraction for each backend module.
- Session-expiration UX overlay when backend returns unauthorized.

5.3 Role-Based Access

- User role is fetched from backend profile endpoint.
- Admin-only navigation items are conditionally shown in the UI.
- Backend remains the ultimate access-control enforcement via claims and customer authorization filter.

5.4 Backend Integration

- Axios client centralizes base URL and token injection.
- Service calls are customer-scoped where applicable using selected customer API identifier.
- 401 responses trigger session-expired handling and user re-authentication flow.

6. Backend Services

6.1 API Design

Two APIs with distinct concerns:

1. Mobile API:
    - Route pattern anchored on customer API identifier for member data and actions.
    - Session token middleware for backend-issued tokens.
2. Management API:
    - Mixed global and customer-scoped routes.
    - Entra bearer token validation through Microsoft Identity integration.
    - Customer authorization filter on customer-scoped controllers.

6.2 Business Logic

- User onboarding and membership state transitions.
- Loyalty level and perk definition and usage constraints.
- Membership pause scheduling and history.

- Notice targeting by membership status and loyalty level.
- QR visit type lifecycle, print payload generation, and visit logs.
- Customer integrations and Trybe-linked booking/membership synchronization.

6.3 Authentication and Authorization

- Mobile channel:
    1. Entra token validation for exchange only.
    2. Backend JWT access and refresh tokens for operational calls.
    3. Refresh rotation and session revocation support.
- Web channel:
    1. Entra tokens validated in Management API.
    2. Claim-derived identity mapped to internal user and role checks.
- Authorization layers:
    1. Global admin role checks.
    2. Customer admin checks in user-customer mapping.
    3. Customer route context validation.

6.4 Database Schema and Data Model

Core model groups include:

- Identity and tenancy:
    1. User.
    2. Customer.
    3. UserCustomer.
    4. UserSession.
- Loyalty domain:
    1. LoyaltyLevel.
    2. LoyaltyLevelPerk.
    3. PerkUsageLog.
    4. UserLoyaltyLevelMembership.
    5. UserMembershipPause.
- Communications and engagement:
    1. Notice.
    2. NoticeAudience.

3. QrVisitType.
4. QrVisitLog.

- Integration and governance:
  1. Integration.
  2. CustomerIntegration.
  3. AuditLog.

The DbContext applies soft-delete query filters and indexes for common tenant and time-based access patterns.

6.5 External Integrations

- Entra External ID for identity and token acquisition.
- Trybe external APIs for customer booking and membership synchronization.
- Integration plugin abstraction exists for lifecycle hooks tied to enabled customer integrations.

7. Infrastructure and DevOps

7.1 Hosting Model

- Backend APIs deployed to Azure App Service as independent applications.
- Web portal deployed to Azure Static Web Apps.
- Data persistence on SQL Server endpoint configured per environment.

7.2 CI/CD Pipelines

- Backend workflow:
  1. Restore, build, test, coverage.
  2. Publish artifacts.
  3. Deploy to environment-specific App Service targets based on branch.
- Web workflow:
  1. Install, type-check, lint, build.
  2. Deploy to dev or prod Static Web App by branch.
- Mobile workflows:
  1. CI analysis/testing workflow available.
  2. Android build and deploy workflows with flavor/customer matrix and signing secrets.
  3. iOS build/TestFlight workflows with customer matrix and signing secrets.
- Security automation:
  1. CodeQL workflow present.

2. Dependabot auto-merge workflow configured for semver patch/minor path.

7.3 Environments

- Branch-based dev and prod deployment targeting is implemented in backend and web workflows.
- Mobile workflows are largely manual-dispatch driven with customer-specific matrix values.

7.4 Monitoring, Logging, and Observability

- Application logging is configured through Serilog to console output.
- Health endpoints are exposed by both APIs.
- Static analysis security scanning is configured via CodeQL.

7.5 Scalability Strategy

Current implementation supports scale-out through stateless API deployment patterns, SQL indexing, and clear API separation between member and admin workloads. Integration and booking provider abstractions support incremental expansion without restructuring core APIs.

8. Security Model

8.1 Authentication

- Mobile:
  1. Entra OTP bootstrap.
  2. Backend-issued access and refresh token pair.
  3. Refresh rotation with one-time-use semantics in session service.
- Web:
  1. Entra token acquisition via MSAL.
  2. Backend token validation in Management API.

8.2 Authorization

- Role-based checks in database-backed user model.
- Customer-level admin enforcement via filter and customer linkage.
- Route-based customer scoping through middleware.

8.3 Data Protection

- Token hashing used for session token verification fields.
- HTTPS redirection configured in runtime pipeline.
- Soft-delete model helps preserve auditability of business records.
- QR payload signature mechanism prevents unauthenticated QR tampering.

8.4 Secrets Management

- GitHub Actions uses repository secrets for Azure deployment credentials and mobile signing assets.
- App configuration includes placeholders and development values; production secret hardening is via secure environment configuration using Azure Key Vault.

8.5 Compliance Considerations

- Customer isolation is enforced at request-routing and query-filtering layers.
- Audit log entity and mutation logging services are present for management operations.