

5. APPLICATION COMPONENTS AND MODULES

The Immigration Application is divided into specialized modules to handle the complex requirements of border management.

5.1 Authentication & Authorization Module

Handles secure login via OAuth2. Supports **Password Grant** for user UI login and **Client Credentials** for secure inter-service API calls.

- **Core Files:** OAuth2AuthorizationConfig.java, AuthController.java
- **Logic:** JWT token issuance, refresh logic, and BCrypt credential verification.

5.2 Immigration Form Management Module

The system's heart, managing the digital "Arrival/Departure" cards. Features deep validation for passport data, flight details, and travel purposes.

- **Core Files:** ImmigrationFormDetailEntity.java, ImmigrationFormFactory.java
- **Logic:** Conditional field validation based on Route (Air/Road/Sea).

5.3 User Management Module

Granular RBAC for Super Admins, Staff, and Agents. Allows city-based assignment for border agents.

5.4 Blacklist Management Module

Security filter that stores passport numbers of restricted individuals. Applications are cross-referenced during submission.

5.5 Form Review & Verification

Agent-side interface for scanning QR codes and marking travelers as "Arrived" or "Departed".

5.6 Ticket Generation

Thymeleaf-based PDF engine creating secure tickets with embedded QR/Barcodes.

5.7 Statistics & Reporting

Aggregates transactional data into dashboard metrics and downloadable Excel reports.

5.8 Email Notifications

Asynchronous engine for sending credentials and traveler tickets.

5.9 Master Data

Global registry for Nationalities, States, and Cities (init via GeoJSON).

5.10 Cron Job Module

Background scheduler for system maintenance and email queue processing.

5.11 API Gateway Module

Centralized routing with CORS management and Swagger UI aggregation for the entire microservice cluster.

6. API DOCUMENTATION

All APIs are routed through the Gateways (Ports 9881/9882) and require appropriate OAuth2 scopes.

6.1 Authentication APIs

POST /oauth/token

Generates access/refresh tokens. Requires Basic Auth (Client ID/Secret).

6.2 Immigration Form APIs

POST /app/public/immigration/api/submit-form

Public endpoint for traveler submission. Returns unique reference code.

GET /app/public/immigration/api/get-form-details

Retrieves form data using Reference Code and Passport Number.

6.3 User Management APIs

ENDPOINT	METHOD	PERMISSION REQUIRED
/admin/secured/api/agent-user-management/agentadminuser	POST	ROLE_USER_MANAGEMENT
/admin/secured/api/agent-user-management/getAgentUserList	POST	ROLE_STAFF

6.4 Blacklist Management APIs

POST /admin/secured/api/blacklist-travellers/add

Adds a passport number to the restricted database.

6.5 Form Review APIs (Agent)

POST /agent/secured/api/mark-status

Updates traveler status to "ARRIVED" or "DEPARTED".

6.6 Statistics & Master Data APIs

- **Dashboard Stats:** POST /admin/secured/api/statistics/dashboard
- **Nationality List:** GET /app/public/master/nationality-list

6.8 Ticket APIs

GET /app/public/ticket/download

Returns a PDF byte stream of the traveler ticket.

7. DATA ARCHITECTURE

The system utilizes **Oracle Database 18c** with a two-schema design to separate security credentials from transactional business data.

7.1 Schema Separation

- **c##immiuserAuth:** Stores OAuth2 tokens (access/refresh), client details, and user credentials.
- **c##immiuserApp:** Stores all immigration forms, blacklist records, and master location data.

7.2 Key Entity Relationships

The architecture emphasizes data integrity via strict foreign key constraints:

- **Form ↔ City:** Each submission links to a City Entity for accommodation tracking.
- **Form ↔ Agent:** Verified forms link to the specific Agent ID who scanned the QR code.
- **City ↔ State:** Hierarchical location mapping for reporting by region.

7.3 Immigration Form Details

CATEGORY	PRIMARY FIELDS
Personal	Names, DOB, Passport No, Nationality, Gender, Marital Status.
Travel	Route (Air/Sea/Road), Flight No, Company, Port of Entry.
Health/Customs	Declarations (Money, Plants, Animals), Symptoms tracking.
Audit	Form Reference Code, Review Status, Reviewed By, Timestamp.

7.4 Indexing & Validation Strategy

Indexing for Performance

- **Unique Index:** FormReferenceCode for instant ticket retrieval.
- **Search Index:** PassportNumber (App Table) and Email (Auth Table).
- **Security Index:** PassportNumber on Blacklist table for real-time submission checks.

Multi-Layer Validation

1. **JSR-303 (Bean Validation):** Enforces `@NotNull` and `@Email` constraints at the controller level.
2. **Service Layer Logic:** Validates age-specific logic (e.g., Minors must have guardian data).
3. **Database Constraints:** Enforces uniqueness and referential integrity via Oracle constraints.

8. SECURITY ARCHITECTURE

Security is the system's core pillar, implementing a defense-in-depth approach across all layers.

8.1 Authentication & Token Management

- **Method:** OAuth2 2.0 with JWT (JSON Web Tokens).
- **Token Storage:** Tokens are stored in the `oauth_access_token` table, allowing for centralized revocation and audit.
- **Encryption:** All user passwords are hashed using **BCrypt** with a high salt cost.

8.2 Authorization & Role Management (RBAC)

Permissions are assigned via Authorities attached to the JWT:

- **ROLE_SUPER_ADMIN:** Universal access.
- **ROLE_AGENT:** Access to scan and review APIs only.
- **ROLE_BLACKLIST_USER_MGMT:** Access restricted to security staff.

8.3 API Protection & Endpoint Security

- **Scope Guarding:** Gateways verify that the token has either `admin_server` or `agent_server` scopes.
- **Rate Limiting:** Public submission endpoints are protected by **Resilience4j** to prevent DDoS or brute-force form submissions.
- **CORS:** Restricted to authorized domains via Gateway configuration.

8.4 Sensitive Data Protection (PII)

PII (Personally Identifiable Information) such as Passport Numbers and Addresses are handled with strict care:

- **Encryption in Transit:** TLS/SSL is required for all API communications.
- **Logging:** The system is configured to avoid logging PII in standard application logs.
- **Access Control:** Database access is restricted to application service accounts with minimal required privileges.

8.5 Audit and Logging

All sensitive operations (Blacklist additions, User creation, Status changes) are logged with a **Trace ID** provided by Spring Cloud Sleuth, allowing developers to trace a single request across multiple microservices.