

# SYSTEM TECHNICAL DOCUMENTATION

Immigration Management System

---

**Document Version:** 1.0

**Client:** Immigration Department

**Project Code:** WMJ16414

**Status:** Final Release

# 1. SYSTEM OVERVIEW

The Immigration Management System (IMS) is an enterprise-grade digital platform engineered to modernize border control operations through automation, real-time data processing, and robust security protocols. It serves as a centralized hub for managing the end-to-end lifecycle of traveler entry and departure.

## 1.1 Business Challenges Addressed

- **Manual Bottlenecks:** Replaces inefficient paper-based arrival/departure cards with a high-speed digital submission engine.
- **Border Security:** Mitigates security risks by integrating a real-time "Blacklist" verification system that checks every traveler against restricted databases instantly.
- **Operational Visibility:** Provides government administrators with live analytics on traveler volume, nationality trends, and port traffic patterns.
- **Data Integrity:** Centralizes fragmented immigration records into a unified, secure Oracle-backed data warehouse.

## 1.2 Target User Personas

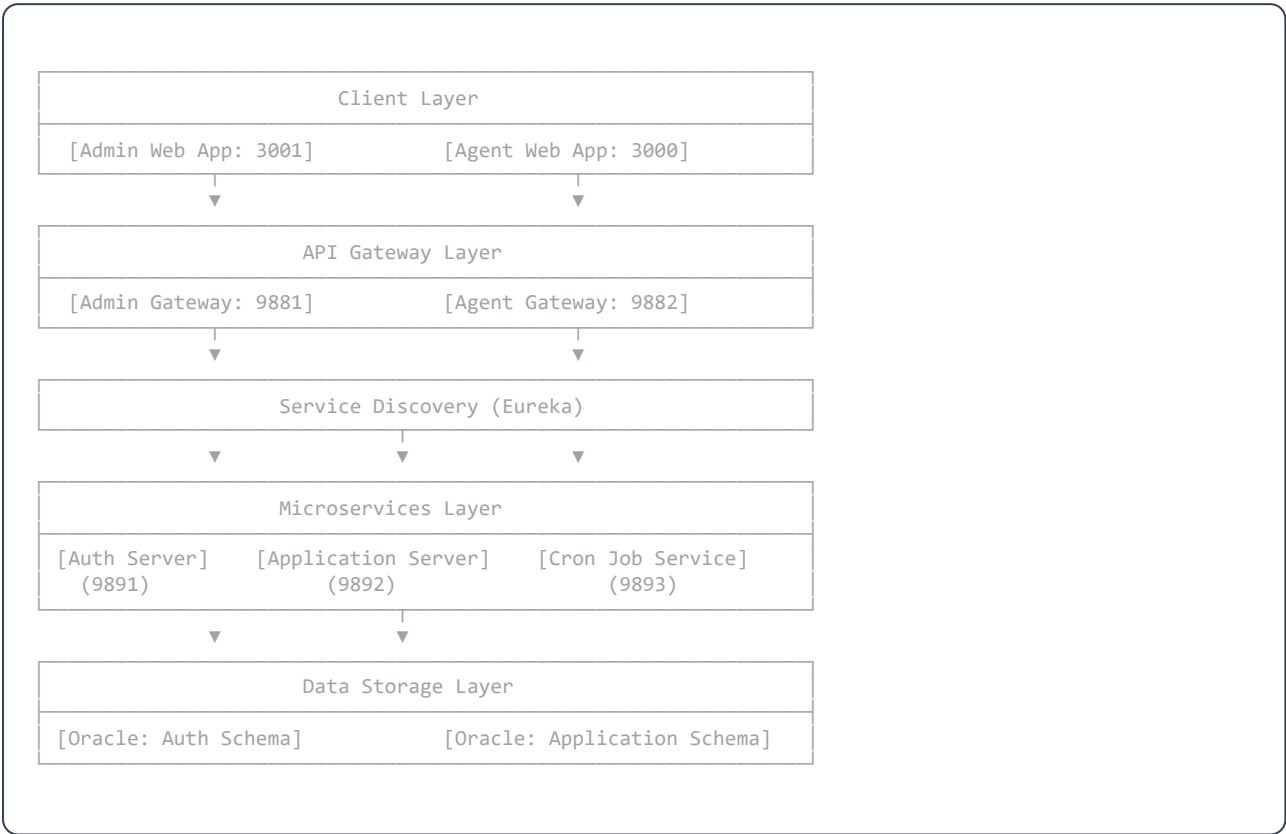
| USER TYPE     | CORE RESPONSIBILITIES                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------|
| Travelers     | Submit digital arrival/departure forms, provide health/goods declarations, and download QR-coded entry tickets.     |
| Border Agents | Utilize scanner interfaces to verify traveler QR codes, review submission details, and finalize entry/exit status.  |
| Admin Staff   | Manage user accounts (Agents/Staff), review flagged applications, and maintain location master data (Cities/Ports). |
| Super Admins  | Complete system oversight, security configuration, audit log analysis, and high-level statistics reporting.         |

## 1.3 Core Capabilities

- **Smart Form Submission:** Logic-driven forms that adapt based on travel route (Air, Land, Sea) and age (Guardian logic for minors).
- **Secure Verification:** QR and Barcode generation for tickets, enabling contactless verification at ports of entry.
- **Automated Notifications:** Real-time email delivery for credentials and travel tickets via background workers.
- **Dynamic Reporting:** On-demand generation of Excel and PDF reports for immigration traffic and security audits.

## 2. HIGH LEVEL ARCHITECTURE

The system utilizes a **Microservices Architecture** pattern based on the Spring Cloud ecosystem. This ensures independent scaling, high availability, and loose coupling between business domains.



### 2.1 Service Boundaries

| SERVICE NAME                | FUNCTIONAL BOUNDARY & RESPONSIBILITY                                                                                   |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>Config Server (9888)</b> | Acts as the central configuration repository. Provides encrypted properties to all services from a Git-based back-end. |
| <b>Discover (9811)</b>      | Netflix Eureka server for service registration, heartbeating, and client-side load balancing.                          |
| <b>Authorization (9891)</b> | OAuth2 Server. Manages user authentication, JWT generation, and client-credential security for inter-service calls.    |
| <b>Application (9892)</b>   | The core business engine. Manages forms, blacklist logic, master data, PDF generation, and dashboard statistics.       |
| <b>CronJob (9893)</b>       | Scheduled task executor. Handles asynchronous email queuing and periodic data cleanup.                                 |

## 2.2 ARCHITECTURE FLOW

---

The following logical flows describe the interaction between system components during critical business operations.

### Form Submission Flow

Traveler → Public API → Application Service → Database → Email Service → Traveler (Ticket)

The traveler accesses the public submission endpoint via the Application Service. The system validates the input, generates a unique reference code, persists the data to Oracle, and triggers an asynchronous email containing the PDF ticket with a QR code.

### Authentication Flow

User → Gateway → Authorization Service → Database → JWT Token → User

Users authenticate via the gateway. The Authorization service verifies credentials against the Auth Schema, generates a signed JWT containing roles (e.g., ROLE\_AGENT), and returns it. This token is used for subsequent authorized requests.

### Form Review Flow

Agent → Agent Gateway → Application Service → Blacklist Check → Database → Status Update

When an agent scans a QR code, the request passes through the Agent Gateway. The Application service retrieves the form, performs a real-time blacklist check, and updates the status to "ARRIVED" or "DEPARTED" upon confirmation.

### Statistics & Reporting Flow

Admin → Admin Gateway → Application Service → Data Analytics → Dashboard/Excel

Administrative users request dashboard metrics. The application service performs aggregation queries on the transactional data, calculates trends (Previous month comparison), and serves the data as JSON for the UI or as a downloadable Excel file.

### 3. TECHNOLOGY STACK

---

The IMS utilizes a modern, robust stack designed for high throughput and enterprise security requirements.

#### Backend & Cloud Frameworks

| CATEGORY      | TECHNOLOGY   | ROLE IN SYSTEM                                        |
|---------------|--------------|-------------------------------------------------------|
| Language      | Java         | Core programming language (Standard for reliability). |
| Framework     | Spring Boot  | Foundation for all microservices logic.               |
| Microservices | Spring Cloud | Distributed configuration, discovery, and routing.    |
| Security      | OAuth2 / JWT | Stateless authentication and role-based access.       |

#### Data, Storage & Utilities

| CATEGORY   | TECHNOLOGY      | ROLE IN SYSTEM                                       |
|------------|-----------------|------------------------------------------------------|
| Database   | Oracle Database | Primary storage for transactional and security data. |
| ORM        | Spring Data JPA | Mapping Java objects to Oracle tables (Hibernate).   |
| PDF Engine | wkhtmltopdf     | Renders HTML/CSS templates into PDF tickets.         |
| DevOps     | Docker          | Containerization for environment consistency.        |

## 4. REPOSITORY STRUCTURE

---

The project follows a **Maven Multi-Module** design to enforce separation of concerns and maximize code reuse across the microservices ecosystem.

```
ImmigrationBackend/ └─ ImmigrationParent/ # Master POM (Dependencies & Versions) └─
ImmigrationProperties/ # Shared Property Libraries └─ ImmigrationCore/ # Core Framework
(Shared Logic) | └─ call_api/ # External API Integration Utils | └─ config/ # Common Security
Configs | └─ exception/ # Unified Error Handling | └─ model/ # Global Data Models └─
ImmigrationConfigServer/ # Distributed Config Management └─ ImmigrationDiscover/ # Eureka
Service Registry └─ ImmigrationAuthorization/ # OAuth2 Auth & User Security └─
ImmigrationApplication/ # Main Business & Public API | └─ controller/ # REST Endpoints | └─
factory/ # Business Logic Factories | └─ pdfgenerator/ # Ticket Generation Engine └─
ImmigrationCronJob/ # Background Task Runner └─ ImmigrationGatewayAdmin/ # Entry point for
Admin UI └─ ImmigrationGatewayAgent/ # Entry point for Agent UI └─ DockerDir/ # Docker
Deployment Orchestration
```

### 4.1 Directory Purpose Explanation

- **ImmigrationCore:** A shared library dependency used by all microservices. It contains the `Validation.java` engine for sanitizing traveler input and `InValidInputFormatException` for standardized error reporting.
- **ImmigrationProperties:** Houses the `application.properties` files used during development and specialized shared configuration strings that are common across environments.
- **ImmigrationApplication:** The largest module containing the actual immigration domain logic. It uses a **Factory Pattern** in the `factory/` directory to separate complex business rules (like Blacklist checking) from the controllers.
- **ImmigrationAuthorization:** Specifically manages the security context. It connects to the Auth schema and handles user registration, password strength validation (Passay), and token lifecycle.
- **DockerDir:** This directory is structured for orchestration. It contains three distinct ``docker-compose.yml`` files (Config, Microservices, Gateways) to manage the strict startup order required by the Spring Cloud environment.