Web application documentation

System purpose and modules

- **Complaint intake portal:** Public users lodge complaints and upload evidence.
- **Case management:** Internal staff triage, assign, investigate, and track SLA.
- **Determination publishing:** Adjudicator drafts, reviews, and publishes outcomes.
- **Unclaimed benefits search:** Public lookup by name/NationalID and fund.
- **Reporting and analytics:** Dashboards for workload, SLA, outcomes, and trends.
- **Administration:** User, role, permission, reference data management.

Roles and permissions

- **Admin:** Full access to users, roles, reference data, and reports.
- **Case officer:** Create/update complaints, manage assignments, add documents, log actions.
- **Adjudicator:** View case files, create/publish determinations, lock cases post-issue.
- **Public user:** Create complaint, upload documents to own case, view own case status, search unclaimed benefits.

API endpoints (REST)

text

```
POST   /api/auth/login
POST   /api/auth/register          (public complainant portal)
GET    /api/complaints             (filter by status, SLA, assignee)
POST   /api/complaints             (create intake)
GET    /api/complaints/{id}
PUT    /api/complaints/{id}        (update details)
PATCH  /api/complaints/{id}/status   (transition with reason)
POST   /api/complaints/{id}/assign   (assign staff)
GET    /api/complaints/{id}/actions  (action log)
POST   /api/complaints/{id}/documents
```

```
GET   /api/complaints/{id}/documents


POST  /api/determinations        (create)

GET   /api/determinations/{id}

PATCH /api/determinations/{id}/publish


GET   /api/unclaimed-benefits      (search)

POST  /api/unclaimed-benefits      (admin/fund create)

PATCH /api/unclaimed-benefits/{id} (status updates)


GET   /api/reports/sla-breaches

GET   /api/reports/workload

GET   /api/reports/outcomes


GET   /api/admin/users

POST  /api/admin/users

PATCH /api/admin/users/{id}/role

GET   /api/admin/reference/statuses

GET   /api/admin/reference/categories
```

Request/response examples

http

```
POST /api/complaints

Content-Type: application/json


{

  "caseNumber": "OPFA-2025-000123",

  "complainantId": 101,

  "fundId": 33,
  "employerId": 12,
  "categoryCode": "NONPAYMENT",
  "summary": "Retirement payout not made",
  "details": "Submitted documents on ...",
  "preferredOutcome": "Payment of full amount"
}
```

http

```
PATCH /api/complaints/123/status
Content-Type: application/json

{
 "toStatus": "IN_REVIEW",
 "note": "Assigned to case officer",
 "reason": "Initial triage complete"
}
```

Business rules

- **Status transitions:**
    - **NEW → IN_REVIEW → AWAITING_INFO → IN_REVIEW → DETERMINED → CLOSED**
    - **Invalid transitions blocked; every change logged.**
- **Single determination per complaint:** Publish locks complaint to CLOSED unless appeal workflow is introduced.
- **SLA policy:**
    - **NEW:** 5 business days to triage.
    - **IN_REVIEW:** 60–90 days target resolution (configurable).
    - **Breaches:** Flag in reports and notify case officer.

Non-functional requirements

- **Security:** HTTPS, strong password hashing, RBAC, input validation, CSRF protection for web forms, secure file storage.

- **Privacy:** Minimize PII, consent capture, redact sensitive uploads, role-based field visibility.
- **Reliability:** Backups, health checks, retry queues for notifications.
- **Performance:** Pagination, indexed queries, async document uploads/notifications.
- **Scalability:** Stateless API, horizontal scaling, object storage for documents.

Deployment architecture

- **Frontend:**
    - **Stack:** React/Vue with TypeScript; role-based views; file upload with virus scanning hook.
- **Backend:**
    - **Stack:** Spring Boot or Node.js (Express/NestJS); REST API; JWT auth.
- **Database:**
    - **RDBMS:** PostgreSQL/MySQL with daily backups and PITR.
- **Storage:**
    - **Documents:** S3-compatible object storage with signed URLs and lifecycle rules.
- **Ops:**
    - **CI/CD:** GitHub Actions; dev/staging/prod; migrations via Flyway/Liquibase.

UI flow highlights

- **Complainant portal:** Guided intake, progress tracker, document upload, secure messaging.
- **Staff dashboard:** Work queue, SLA alerts, quick filters, recent actions.
- **Determination editor:** Rich text, template snippets, preview, publish toggle.
- **Unclaimed benefits:** Simple search, match suggestions, resolution steps.
- **Reports:** Charts for SLA breaches, outcomes distribution, workload by officer.

Data quality, migration, and testing

Data quality controls

- **Validation:** Required fields, format checks (NationalID, email), deduplication on complainants/funds.
- **Reference data:** Controlled vocabularies via Ref_* tables; no free-form statuses.
- **Document hygiene:** File type whitelist, size limits, malware scan.

Migration approach

- **Staging import:** CSV/Excel loaders for Funds/Employers/UnclaimedBenefits with mapping and validation logs.
- **Idempotency:** Use external identifiers to avoid duplicates.
- **Reconciliation:** Reports for import errors and unmatched records.

Testing matrix

- **Unit tests:** Services, repositories, validators, status transition rules.
- **Integration tests:** API endpoints with RBAC, audit trail creation, file upload flows.
- **Performance tests:** Workload queries, SLA report generation under load.
- **Security tests:** Auth flows, role/fine-grained permission checks, injection attempts.
- **UAT:** Role-specific scripts: Admin, Case Officer, Adjudicator, Public user.