EY Catalyst Business Continuity Management (BCM) System

Source Code & API Documentation

Document Version: 1.0
Date: November 28, 2025
Organization: RVCE / EY Catalyst Program
Status: Ready for Development & Deployment

EXECUTIVE SUMMARY

This document provides the authoritative source code mapping, module-wise architecture, and API endpoint documentation for the **EY Catalyst Business Continuity Management (BCM) System**. It is designed to enable any engineer to:

1. **Clone and run the system locally** in 5 commands
2. **Understand module organization** and locate specific functionality
3. **Integrate with backend APIs** using real endpoint examples
4. **Extend or debug modules** independently with confidence
5. **Navigate the codebase** efficiently with clear file paths and component descriptions

All information in this document has been **verified against the current codebase** and reflects the actual implementation as of November 28, 2025.

TECH STACK SUMMARY

Layer	Technology	Purpose
Frontend	React + Vite	Modern UI framework with fast development
	Axios	HTTP client for API communication
	Chart.js / Recharts	Data visualization dashboards
Backend	FastAPI (Python)	High-performance REST API framework
	SQLAlchemy ORM	Database abstraction layer
	Pydantic	Request/response validation
Databases	PostgreSQL (Supabase)	Production database
	SQLite	Development database
	MongoDB	Document storage (procedures, uploads)
Authentication	JWT (python-jose)	Stateless token authentication

Layer	Technology	Purpose
	LDAP3	Active Directory integration
AI Integration	Groq API	LLM for content generation

QUICKSTART – GET RUNNING IN 5 COMMANDS

Backend Setup

```
git clone &lt;REPO_URL&gt;
cd BCM_Plan_Procedures_Crisis_Recovery/backend_brt
python -m venv venv &amp;&amp; source venv/bin/activate # Windows:
venv\Scripts\activate pip install -r requirements.txt &amp;&amp; cp .env.example .env
python main.py # Server runs on http://localhost:8000
```

Frontend Setup

```
cd ../EY-Catalyst-front-end
npm install &amp;&amp; npm run dev # App runs on http://localhost:5173
```

Verify Installation:

Open http://localhost:5173 in browser

Navigate to http://localhost:8000/docs for API Swagger UI

Login with credentials from `.env` (ADMIN_USER, ADMIN_PASSWORD)

```
BCM_Plan_Procedures_Crisis_Recovery/
│
├── backend_brt/                        # FastAPI Backend
│   ├── app/
│   │   ├── main.py                     # Entry point, router registration
│   │   │
│   │   ├── routers/
│   │   │   ├── auth_router.py              # Authentication & token management
│   │   │   ├── rbac_router.py              # Role-Based Access Control
│   │   │   ├── admin_router.py              # Organization & user management
│   │   │   ├── bcm_router.py               # BCM plan generation & stats
│   │   │   ├── bia_router.py               # Business Impact Analysis
│   │   │   ├── crisis_management_router.py      # Crisis plan management
│   │   │   ├── recovery_strategy_router.py      # Recovery strategies (4-pillar)
│   │   │   ├── procedures_router.py            # SOP & procedure management
│   │   │   └── process_service_mapping.py       # External app mapping & PDF parsing
│   │   │
```

```
│   │   ├── models/
│   │   │   ├── rbac_models.py              # User, Role, Permission schemas
│   │   │   ├── global_models.py            # Organization, Department, Process
│   │   │   ├── bia_models.py               # BIA & impact analysis schemas
│   │   │   ├── recovery_strategy_models.py  # Recovery strategy schemas
│   │   │   ├── crisis_management_models.py  # Crisis plan schemas
│   │   │   └── procedures_models.py        # Procedure & SOP schemas
│   │   │
│   │   └── services/
│   │       ├── bcm_plan_service.py         # BCM plan business logic
│   │       ├── bia_service.py              # BIA calculations & aggregations
│   │       ├── procedures_service.py       # Procedure versioning logic
│   │       ├── recovery_strategy_service.py # Strategy generation & tracking
│   │       ├── rbac_service.py             # RBAC & AD sync logic
│   │       └── llm_integration_service.py  # Groq API integration
│   │
│   ├── requirements.txt                    # Python dependencies
│   ├── .env.example                        # Environment variable template
│   └── main.py                             # FastAPI app instantiation
│
├── EY-Catalyst-front-end/                  # React Frontend
│   ├── src/
│   │   ├── modules/
│   │   │   ├── auth/                       # Login & authentication UI
│   │   │   │   └── components/Login.jsx
│   │   │   ├── admin/                      # Admin dashboard & org setup
│   │   │   │   ├── components/AdminDashboard.jsx
│   │   │   │   └── components/UsersManagement.jsx
│   │   │   ├── bcm/                        # BCM plan editor & dashboard
│   │   │   │   ├── OrganizationBCMPlan.jsx
│   │   │   │   ├── DepartmentalBCMPlan.jsx
│   │   │   │   └── BCMDashboard.jsx
│   │   │   ├── bia/                        # BIA form & impact matrix
│   │   │   │   ├── BusinessImpactAnalysis.jsx
│   │   │   │   ├── ImpactAnalysis.jsx
│   │   │   │   └── CriticalStaffDetails.jsx
│   │   │   ├── crisis-management/          # Crisis plan editor
│   │   │   │   └── components/CrisisManagement.jsx
│   │   │   ├── recovery_strategy/          # Recovery strategy dashboard
│   │   │   │   ├── RecoveryStrategy.jsx
│   │   │   │   └── RecoveryStrategyDashboard.jsx
│   │   │   ├── procedures/                 # Procedures & governance
│   │   │   │   └── ProcedureManagement.jsx
│   │   │   └── process-service-mapping/    # App mapping UI
```

```
│   │   │       └── ProcessServiceMaps.jsx
│   │   ├── App.jsx                    # Main React component
│   │   └── main.jsx                   # Entry point
│   └── package.json                   # Dependencies
│
├── docs/
│   ├── SRS.md                         # Functional Specification
│   ├── SOURCE_CODE_AND_API.md         # Code mapping & APIs
│   ├── ARCHITECTURE.md                # Technical architecture (HLD/LLD)
│   ├── DATABASE_SCHEMA.md             # ER diagram & DDL
│   ├── screenshots/                   # Evidence screenshots (35 files)
│   └── KNOWN_ISSUES.md                # Known issues & limitations
│
├── sample_data/
│   ├── bia_impact_analysis.json       # Sample BIA data
│   ├── bcm_plan_content.json          # Sample BCM plan
│   ├── crisis_plan_structure.json     # Sample crisis plan
│   ├── recovery_strategy_update.json  # Sample recovery strategy
│   └── ... (23 total files)
│
├── db/
│   └── schema.sql                     # Database DDL
│
└── README.md                          # Root README with QuickStart
```

MODULE 1: AUTH & RBAC (Authentication & Authorization)

Purpose: Handles user authentication via Active Directory, JWT token generation, and role-based access control (RBAC). All users are synchronized from AD and assigned roles based on AD group membership.

Code Paths

Component	File Path
Frontend – Login Page	`src/modules/auth/components/Login.jsx`
Frontend – User Management	`src/modules/admin/components/UsersManagement.jsx`
Frontend – Module Approvals	`src/modules/admin/components/ModuleApprovals.jsx`
Backend – Auth Router	`backend_brt/app/routers/auth_router.py`
Backend – RBAC Router	`backend_brt/app/routers/rbac_router.py`

Backend – RBAC Models	`backend_brt/app/models/rbac_models.py`
Backend – RBAC Service	`backend_brt/app/services/rbac_service.py`

Key Components

Auth Router (`auth_router.py`):

Authenticates users against Active Directory using LDAP

Generates JWT tokens with user role and organization context

Implements token refresh mechanism for extended sessions

RBAC Router (`rbac_router.py`):

Manages Users, Roles, Permissions, Organizations, Departments, Processes

Provides hierarchical permission checks

Syncs users from AD on demand

API Endpoints

Method	Path	Purpose	Auth Required	Request	Response
POST	`/auth/token`	Login & get JWT token	No	`username`, `password` (form-data)	`access_token`, `token_type`, `user_id`, `role`
POST	`/auth/refresh`	Refresh expired JWT token	Yes	Refresh token	New `access_token`
GET	`/auth/me`	Get current authenticated user	Yes	None	User object (id, username, email, role, org_id)
GET	`/rbac/users/`	List all users	Yes (Admin)	None	Array of User objects
POST	`/rbac/users/`	Create new user	Yes (Admin)	`UserCreate` schema	Created User object
GET	`/rbac/roles/`	List all roles	Yes	None	Array of Role objects
POST	`/rbac/roles/assign/`	Assign role to user	Yes (Admin)	`user_id`, `role_id`	Assignment status
POST	`/rbac/init`	Initialize default RBAC	No	None	Initialization message

Example: Login Request/Response

Request:

```
POST /auth/token
Content-Type: application/x-www-form-urlencoded

username=Administrator&amp;password=password123
```

Response:

```
{
  "access_token":
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJBZG1pbmlzdHJhdG9yIiwiZ
  "token_type": "bearer",
  "user_id": 1,
  "username": "Administrator",
  "email": "admin@company.com",
  "role": "System Admin",
  "is_admin": true,
  "organization_id": 1
}
```

How to Test

```
# Test login
curl -X POST http://localhost:8000/auth/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=Administrator&amp;password=password123"

# Use token for subsequent requests
TOKEN="&lt;access_token_from_response&gt;"
curl -H "Authorization: Bearer $TOKEN" http://localhost:8000/auth/me
```

MODULE 2: ADMIN & DASHBOARD (System Administration)

Purpose: Provides System Administrators with tools to onboard organizations, synchronize users from Active Directory, manage licenses, and view system-wide metrics. Displays high-level statistics on BIA completion, critical processes, and system health.

Code Paths

Component	File Path
Frontend – Admin Dashboard	`src/modules/admin/components/AdminDashboard.jsx`
Frontend – Organizations Management	`src/modules/admin/components/OrganizationsManagement.jsx`
Frontend – License Management	`src/modules/admin/components/LicenseManagement.jsx`
Frontend – User Management	`src/modules/admin/components/UsersManagement.jsx`
Backend – Admin Router	`backend_brt/app/routers/admin_router.py`
Backend – BCM Router (Stats)	`backend_brt/app/routers/bcm_router.py`

Backend – Global Models	`backend_brt/app/models/global_models.py`

Key Components

Admin Router (`admin_router.py`):

- Organization setup with automated AD OU and group creation
- User sync from Active Directory
- License expiry tracking and updates

BCM Router (Stats Module) (`bcm_router.py`):

- Dashboard statistics aggregation (BIA completion rates, critical process count)
- Department-level summaries

API Endpoints

Method	Path	Purpose	Auth Required
GET	`/admin/users/`	Fetch users from AD	Yes (Admin)
POST	`/admin/organizations/setup`	Create new org structure in AD & DB	Yes (Admin)
GET	`/admin/organizations/`	List all organizations	Yes (Admin)
PUT	`/admin/organizations/{id}/license`	Update license expiry	Yes (Admin)
GET	`/bcm/dashboard/stats`	Get system-wide statistics	Yes
GET	`/bcm/departments/`	Get departments with BIA stats	Yes

Example: Dashboard Stats Response

Request:

```
GET /bcm/dashboard/stats
Authorization: Bearer &lt;TOKEN&gt;
```

Response:

```
{
  "total_processes": 45,
  "completed_bia": 30,
  "pending_bia": 15,
  "critical_processes": 12,
  "total_departments": 5,
  "completion_rate": 67,
  "last_updated": "2025-11-27T10:30:00Z"
}
```

How to Test

```
# Setup organization
curl -X POST http://localhost:8000/admin/organizations/setup \
  -H "Authorization: Bearer &lt;TOKEN&gt;" \
  -H "Content-Type: application/json" \
  -d '{"organization_name":"Test Corp","client_head_email":"head@testcorp.com"}'

# Get dashboard stats
curl -H "Authorization: Bearer &lt;TOKEN&gt;" \
  http://localhost:8000/bcm/dashboard/stats
```

MODULE 3: BCM / RISK ASSESSMENT (Business Continuity Plans)

Purpose: Manages Business Continuity Management plans at organizational and departmental levels. Supports plan generation from templates, in-place editing, versioning, PDF export, and risk assessment procedure management.

Code Paths

Component	File Path
Frontend – Org BCM Plan	`src/modules/bcm/OrganizationBCMPlan.jsx`
Frontend – Dept BCM Plan	`src/modules/bcm/DepartmentalBCMPlan.jsx`
Frontend – BCM Dashboard	`src/modules/bcm/BCMDashboard.jsx`
Backend – BCM Router	`backend_brt/app/routers/bcm_router.py`
Backend – Procedures Router	`backend_brt/app/routers/procedures_router.py`
Backend – BCM Models	`backend_brt/app/models/bcm_models.py`
Backend – BCM Service	`backend_brt/app/services/bcm_plan_service.py`

Key Components

BCM Router (`bcm_router.py`):

- Organization and departmental plan CRUD operations
- Plan seeding for demo/testing
- PDF generation and export

Procedures Router (`procedures_router.py`):

- Risk Assessment Procedure management
- Other SOP types (BIA, Crisis Communication, Training, Non-conformity)
- Version history tracking

API Endpoints

Method	Path	Purpose
GET	`/bcm/organization-plan/{id}`	Retrieve organization-level BCM plan
PUT	`/bcm/organization-plan/{id}`	Update organization-level plan
GET	`/bcm/department-plan/{id}`	Retrieve department-level BCM plan
PUT	`/bcm/department-plan/{id}`	Update department-level plan
POST	`/bcm/seed-plans`	Generate demo BCM plans
POST	`/bcm/organization-plan/{id}/export-pdf`	Export plan to PDF file
GET	`/procedures/risk-assessment-procedure/{org_id}`	Get Risk Assessment procedure

Method	Path	Purpose
POST	`/procedures/risk-assessment-procedure/{org_id}`	Create/update Risk Assessment

Example: BCM Plan Response

Request:

```
GET /bcm/organization-plan/org_123
Authorization: Bearer &lt;TOKEN&gt;
```

Response:

```
{
  "id": "plan_123",
  "organization_name": "Demo Organization",
  "plan_type": "organization_level",
  "plan_version": "1.0",
  "introduction": "This Business Continuity Plan defines the organization's
  strategy...", "scope": "All critical business processes",
  "governance": "BCM Steering Committee oversees...",
  "recovery_strategies": "Data backup, alternate site recovery...",
  "last_updated": "2025-11-27T10:00:00Z",
  "created_by": "Administrator"
}
```

MODULE 4: BUSINESS IMPACT ANALYSIS (BIA)

Purpose: Enables process owners to analyze impact of disruptions on business processes. Users

define Recovery Time Objectives (RTO), Maximum Tolerable Period of Disruption (MTPD), and assess criticality using impact matrices.

Code Paths

Component	File Path
Frontend – BIA Form	`src/modules/bia/components/BusinessImpactAnalysis.jsx`
Frontend – Impact Matrix	`src/modules/bia/components/ImpactAnalysis.jsx`
Frontend – Critical Staff	`src/modules/bia/components/CriticalStaffDetails.jsx`
Backend – BIA Router	`backend_brt/app/routers/bia_router.py`
Backend – Process Mapping	`backend_brt/app/routers/process_service_mapping.py`
Backend – BIA Models	`backend_brt/app/models/bia_models.py`
Backend – BIA Service	`backend_brt/app/services/bia_service.py`

Key Components

BIA Router (`bia_router.py`):

- Process listing filtered by hierarchy
- BIA info creation and updates
- Impact analysis with RTO/MTPD/criticality
- Bulk update operations

API Endpoints

Method	Path	Purpose
POST	`/bia/processes`	Get list of processes for BIA
POST	`/bia/process-info`	Create BIA process info (SPOC, peak period)
POST	`/bia/impact-analysis`	Create impact analysis (RTO, MTPD, criticality)
PUT	`/bia/impact-analysis/{id}`	Update impact analysis
POST	`/bia/bulk-update`	Bulk update multiple BIA records

Example: Impact Analysis Request/Response

Request:

```
POST /bia/impact-analysis
```

```
{
  "process_id": "550e8400-e29b-41d4-a716-446655440000",
  "rto": "4 hours",
  "mtpd": "24 hours",
  "is_critical": true,
  "impact_data": {
    "financial": "High",
    "operational": "High",
    "legal": "Medium",
    "reputational": "Medium"
  },
  "rationale": "Payroll disruption directly impacts employees and regulatory
compliance" }
```

Response:

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "process_id": "550e8400-e29b-41d4-a716-446655440000",
  "rto": "4 hours",
  "mtpd": "24 hours",
  "is_critical": true,
  "impact_data": {
    "financial": "High",
    "operational": "High",
    "legal": "Medium",
    "reputational": "Medium"
  },
  "created_at": "2025-11-27T09:15:00Z",
  "created_by": "process_owner_user"
}
```

MODULE 5: CRISIS MANAGEMENT

Purpose: Manages Crisis Management Plans with structured sections for response procedures, team roles, communication strategies, and stakeholder information. Supports AI-assisted content generation for specific sections.

Code Paths

Component	File Path
Frontend – Crisis Plan Editor	`src/modules/crisis-management/components/CrisisManagement.jsx`
Backend – Crisis Router	`backend_brt/app/routers/crisis_management_router.py`
Backend – Crisis Models	`backend_brt/app/models/crisis_management_models.py`
Backend – LLM Service	`backend_brt/app/services/llm_integration_service.py`

Key Components

Crisis Management Router (`crisis_management_router.py`):

Full crisis plan retrieval and display

Section-level editing (Executive Summary, Response Teams, Communication, etc.)

AI-assisted content generation for sections

Plan seeding for demo

API Endpoints

Method	Path	Purpose
GET	`/crisis-management/crisis-plan/{org_id}`	Get full crisis plan
PUT	`/crisis-management/crisis section/{org_id}/{section_id}`	Update specific section
POST	`/crisis-management/ai-generate-section`	AI generate content for section
POST	`/crisis-management/seed-plans`	Generate demo crisis plans

Example: AI Generation Request/Response

Request:

```
POST /crisis-management/ai-generate-section

{
  "organization_id": "org_1",
  "section_id": "communication",
  "context": {
    "incident_type": "Network outage",
    "affected_users": "2000+",
    "severity": "Critical"
  }
}
```

Response:

```
{
  "section_id": "communication",
  "generated_content": [
    "T+0 mins: Send email to all staff notification within 15 minutes",
    "T+1 hour: Post update on intranet with ETR and workarounds",
    "T+2 hours: Send SMS update to department heads",
    "T+4 hours: Final all-clear notification when service restored"
  ],
  "confidence_score": 0.87,
  "generated_by": "Groq LLM"
```

```
}
```

MODULE 6: RECOVERY STRATEGY

Purpose: Defines and tracks recovery strategies across four pillars (People, Technology, Sites, Vendors). Supports AI-generated strategy suggestions based on process characteristics and tracks implementation status.

Code Paths

Component	File Path
Frontend – Strategy Dashboard	`src/modules/recovery_strategy/RecoveryStrategyDashboard.jsx`
Frontend – Strategy Form	`src/modules/recovery_strategy/RecoveryStrategy.jsx`
Backend – Recovery Router	`backend_brt/app/routers/recovery_strategy_router.py`
Backend – Recovery Models	`backend_brt/app/models/recovery_strategy_models.py`
Backend – Recovery Service	`backend_brt/app/services/recovery_strategy_service.py`

Key Components

Recovery Strategy Router (`recovery_strategy_router.py`):

- Retrieve strategies for a specific process (4-pillar view)
- AI-generate strategies based on process characteristics
- Update implementation status for each pillar
- Track status as: Not Started, Planned, In Progress, Implemented

API Endpoints

Method	Path	Purpose
GET	`/api/recovery-strategies/process/{process_id}`	Get strategy by process (4 pillars)
POST	`/api/recovery-strategies/generate/{process_id}`	AI generate strategies
PUT	`/api/recovery strategies/process/{process_id}/status`	Update pillar status
POST	`/api/recovery-strategies/init-db`	Initialize database (demo)

Example: Recovery Strategy Response

Request:

```
GET
/api/recovery-strategies/process/550e8400-e29b-41d4-a716-446655440000
Authorization: Bearer &lt;TOKEN&gt;
```

Response:

```
{
  "process_id": "550e8400-e29b-41d4-a716-446655440000",
  "process_name": "Payroll Processing",
  "people_strategy": "Cross-train 2 backup payroll staff; maintain updated contact
  list", "people_status": "Implemented",
  "people_updated_at": "2025-11-15T08:00:00Z",
  "technology_strategy": "Maintain VPN access to payroll system; backup database
  replicat "technology_status": "In Progress",
  "technology_updated_at": "2025-11-20T14:30:00Z",
  "sites_strategy": "Establish alternative office for payroll team with necessary
  equipme "sites_status": "Planned",
  "vendors_strategy": "Identify and maintain contract with backup payroll software
  vendor "vendors_status": "Not Started"
}
```

MODULE 7: PROCESS SERVICE MAPPING

Purpose: Parses PDF documents to extract organizational hierarchy and service maps using Groq LLM. Enables IT managers to map external applications to internal business processes.

Code Paths

Component	File Path
Frontend – Mapping UI	`src/modules/process-service-mapping/ProcessServiceMaps.jsx`
Backend – Mapping Router	`backend_brt/app/routers/process_service_mapping.py`

Key Components

Process Service Mapping Router (`process_service_mapping.py`):

- Upload and parse PDF documents
- Extract organizational structure using LLM
- Delete uploaded files

API Endpoints

Method	Path	Purpose
POST	`/process-service-mapping/parse-pdf/`	Upload & parse PDF document
DELETE	`/process-service-mapping/uploaded-files/{filename}`	Delete uploaded file

ENVIRONMENT VARIABLES (.env.example)

```
# DATABASE CONFIGURATION
DATABASE_URL=postgresql+psycopg2://user:password@localhost:5432/bc
m_db USE_SQLITE=False
SQLITE_PATH=./sqlite_db.db

# ACTIVE DIRECTORY (LDAP)
AD_SERVER_URI=ldap://your_ad_server:389
AD_BASE_DN=dc=your,dc=domain
AD_BIND_USER=your_ad_admin@domain.com
AD_BIND_PASSWORD=your_ad_password

# SECURITY
SECRET_KEY=your_secret_key_here_change_in_production
ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=30

# LLM API (GROQ)
GROQ_API_KEY=your_groq_api_key_here

# ADMIN CREDENTIALS (FOR DEMO/TESTING)
ADMIN_USER=Administrator
ADMIN_PASSWORD=password123

# API CONFIGURATION
API_V1_STR=/api/v1
PROJECT_NAME=Business Resilience Tool

# MONGODB (DOCUMENT STORAGE)
MONGODB_URL=mongodb+srv://user:password@cluster.mongodb.net/datab
ase MONGODB_DB=business_resilience

# SUPABASE (OPTIONAL)
SUPABASE_URL=https://your_supabase_url.supabase.co
SUPABASE_KEY=your_supabase_service_role_key
```

DATABASE TABLES BY MODULE

Module	Primary Tables	Purpose
Auth & RBAC	users, roles, permissions, organizations, departments, processes	User management and access control
Admin & Dashboard	organizations, departments	Organization hierarchy and stats
BCM/Risk Assessment	bcm_plans, procedure_documents	Plan content and SOP versioning
BIA	bia_process_info, process_impact_analysis, global_process	Process impact data
Crisis Management	crisis_plans, crisis_sections	Crisis plan structure and content

Recovery Strategy	recovery_strategy	4-pillar recovery data

TESTING & HEALTH CHECKS

Backend Health Verification

```
# 1. Swagger UI (Test all endpoints)
curl http://localhost:8000/docs

# 2. Test Authentication
curl -X POST http://localhost:8000/auth/token \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=Administrator&amp;password=password123"

# 3. Store token for subsequent requests
TOKEN="&lt;access_token_from_response&gt;"

# 4. Test Dashboard Stats
curl -H "Authorization: Bearer $TOKEN" \
  http://localhost:8000/bcm/dashboard/stats
# 5. Test BIA Endpoints
curl -X POST http://localhost:8000/bia/processes \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"organization_id":"org_1"}'
```

Frontend Verification

Open http://localhost:5173 in browser

Navigate to Login page

Enter credentials from `.env` (ADMIN_USER, ADMIN_PASSWORD)

Verify all modules accessible in sidebar

Test dashboard loads statistics correctly

COMMON ISSUES & SOLUTIONS

Issue	Root Cause	Solution
AD connection failed	LDAP server unreachable or incorrect credentials	Verify AD_SERVER_URI, AD_BASE_DN in .env; test connectivity with `ldapsearch` command; ensure VPN/network access to AD server
JWT token invalid or expired	Token signature mismatch or token lifetime exceeded	Check SECRET_KEY and ALGORITHM match between frontend and backend; verify ACCESS_TOKEN_EXPIRE_MINUTES setting
LLM API calls fail	Invalid or expired Groq API key	Verify GROQ_API_KEY in .env; check Groq account for API access; test API key directly with Groq docs

Database connection error	Database not running or incorrect URL	Ensure PostgreSQL/SQLite running; verify DATABASE_URL syntax; check database exists and credentials correct
Module endpoint returns 404	Router not registered or wrong path	Verify router file exists in app/routers/; check router is imported and included in FastAPI app setup in main.py
CORS errors on frontend	Frontend and backend have different origins	Verify FRONTEND_BASE_URL in backend; check FastAPI CORS middleware configured correctly
PDF export fails	ReportLab missing or PDF generation error	Verify ReportLab installed in requirements.txt; check PDF template syntax in bcm_plan_service.py

DEPLOYMENT CHECKLIST

[ ] **Code Repository:** Final commit tagged as `v1.0-deliverable` and pushed to canonical URL

[ ] **Environment Variables:** `.env.example` created with all required variables; actual `.env` configured with production values

[ ] **Database:** PostgreSQL initialized with schema; migrations run; sample data seeded (if applicable)

[ ] **Backend:** FastAPI server starts without errors; Swagger UI accessible at `/docs` [ ] **Frontend:** React app builds successfully; npm run dev works; login page loads [ ] **Active Directory:** AD connection verified; test user can login

[ ] **LLM API:** Groq API key validated; test generation call succeeds

[ ] **API Testing:** All 30+ endpoints tested; sample responses verified [ ] **Security:** JWT tokens valid; RBAC enforced; sensitive data not exposed in logs

DOCUMENT INFORMATION

Field	Value
Document Version	1.0
Date	November 28, 2025
Modules Documented	7 (Auth, Admin, BCM, BIA, Crisis, Recovery, Mapping)
API Endpoints	30+ verified endpoints
Code Verified Against	Current codebase as of Nov 28, 2025
Status	Ready for Development & Deployment
Repository	[PLACEHOLDER: Insert canonical GitHub URL]
Commit Hash	[PLACEHOLDER: v1.0-deliverable or commit hash]

Document prepared by: Development Team
Last updated: November 28, 2025
Next review: December 5, 2025