**Technical details**

* **Overview of BRM System**
* BRM is a modern .Net Core (.Net 9) Server Application.
* The web pages uses Bootstrap 5 css and does not have any further libraries or third party component dependencies.
* Using .Net Core Hosting Bundle for deployment to use Windows Authentication otherwise the App is platform independent.
* BRM Security is controlled by Active Directory

3. Main Projects & Components

Blazor UI Projects

* **Sassa.Brm**: Main Blazor Server app, user interface, navigation, and business logic.
* **Brm.Fluent**: UI components, dialogs, and additional Blazor features.
* **Sassa.Admin**: Admin UI for office and document management.
* **Sassa.Audit**: Access to Legacy systems datasets (MIS,EMIS,TDW)

API Layer

* **Sassa.Brm.Api**: ASP.NET Core Web API, exposes endpoints for Fasttrack, Application, Scan, and other controllers.
* **Controllers**: FasttrackController, ApplicationController, ScanController, etc.
* **Services**: FasttrackService, ApplicationService, ActivityService, SharedFolderService, etc.

Common Services

* **Sassa.Brm.Common**: Shared logic, helpers, authentication, session management, email, and static data.
* **Helpers**: Security, BulkPrint, WindowsAuthenticationStateProvider, etc.
* **Services**: StaticService, SessionService, EmailClient, MailMessages, UserSettings.

External/Supporting Services

* **Sassa.CsService**: Content Server integration.
* **Sassa.Socpen.Data**: Socpen data access and context.
* **Sassa.Audit**: Audit and reporting services.

Database

* **Oracle Database**: Data storage for BRM, Socpen, and related entities.
* Accessed via Entity Framework Core (ModelContext, SocpenContext).

---

4. API Endpoints – Detailed Flow

4.1 Fasttrack Endpoint

**Route:** **[POST] /Fasttrack**  
**Controller:** FasttrackController  
**Service:** FasttrackService

**Flow:**

* Receives a FasttrackScan object.
* Sets scan folder.
* Checks for the existence of the expected PDF file.
* Loads and validates coversheet data, builds a DcFile object.
* Validates business rules (ID, dates, grant type, etc.).
* Checks for a matching Socpen record.
* Adds or updates the BRM record in the database.
* Updates claim number in LoService.
* Updates Socpen record with BRM info.
* Renames and moves the PDF to the processed folder.
* (Optional) Uploads to Content Server.
* Returns success or error response.

4.2 Application Endpoint

**Route:** **[POST] /Application**  
**Controller:** ApplicationController  
**Service:** ApplicationService

**Flow:**

* Receives an **ApplicationRequest** or similar DTO.
* Validates input (required fields, formats).
* Loads or creates a DcFile or related entity.
* Applies business rules (e.g., grant type, applicant status).
* Saves or updates the application record in the database.
* May trigger additional logic (e.g., notifications, audit logging).
* Returns the created/updated application or error details.

4.3 Scan Endpoint

**Route:** **[POST] /Scan**  
**Controller:** ScanController  
**Service:** **ScanService**

**Flow:**

* Receives a scan request (e.g., barcode, applicant info).
* Validates scan data.
* Locates the corresponding file or record in the database.
* Updates scan status, timestamps, and metadata.
* May move or rename files in the scan folder.
* Returns scan result, status, or error.

4.4 Activity Endpoint

**Route:** **[GET/POST] /Activity**  
**Controller:** ActivityController  
**Service:** ActivityService

**Flow:**

* Receives activity requests (e.g., user actions, audit events).
* Validates activity data.
* Logs activity in the database.
* May trigger notifications or further processing.
* Returns activity log or status.

4.5 Other Endpoints (Office, Document, Socpen)

**Routes:**

* **[GET/POST] /Office**
* **[GET/POST] /Document**
* **[GET/POST] /Socpen**

**Controllers/Services:**

* **OfficeController** / **OfficeService**
* **DocumentController** / **DocumentService**
* **SocpenController** / SocpenService

**Flow:**

* Receive requests for office management, document operations, or Socpen data.
* Validate input and permissions.
* Perform CRUD operations on respective entities.
* Integrate with external systems (e.g., Content Server, Socpen).
* Return results, status, or errors.

---

5. Common Patterns Across Endpoints

* **Validation:** All endpoints validate input data and business rules.
* **Service Layer:** Controllers delegate business logic to services.
* **Database Access:** Entity Framework Core is used for all CRUD operations.
* **Error Handling:** Exceptions are caught, logged, and returned as error responses.
* **Integration:** Endpoints may interact with file system, external APIs, or other services.
* **Response:** Return standardized API responses (success, error, data).

---

6. Key Classes & Data Models

* **DcFile**: Represents a BRM file record, including metadata, status, and links to Socpen.
* **DcSocpen**: Represents a Socpen record, linked by applicant and grant type.
* **CustCoversheet**: Data transfer object for coversheet information.
* **FasttrackScan**: Input model for Fasttrack API.

---

7. Configuration & Environment

* **appsettings.json / appsettings.Production.json**: Stores connection strings, folder paths, service endpoints, and other settings.
* **IWebHostEnvironment**: Used to resolve physical paths for file operations.

---

8. Security & Authentication

* **Windows Authentication**: Used for Blazor Server and API access.
* **Role-based Authorization**: Enforced in UI and API controllers.

---

9. Deployment & Hosting

* **Kestrel/HTTP.sys/IIS**: Supported hosting models.
* **Windows Service**: API can be installed as a Windows Service for auto-start on boot.

---

10. Extensibility & Integration

* **Content Server (CSService)**: Document management integration via SOAP/WCF.
* **Email Notifications**: SMTP settings and mail client for alerts and reports.
* **External APIs**: Socpen, audit, and other government systems.

---

11. Error Logging & Monitoring

* **Serilog**: Used for error and event logging.
* **Health Checks**: Exposed via Blazor UI and API endpoints.

---

12. Best Practices & Recommendations

* **Use Dependency Injection** for all services and data contexts.
* **Validate all input** at API boundaries.
* **Handle exceptions** and log errors for diagnostics.
* **Secure sensitive configuration** (connection strings, credentials).
* **Document all endpoints and models** for maintainability.

---

13. References & Further Reading

* ASP.NET Core Blazor Documentation
* Entity Framework Core Documentation
* Serilog Documentation
* Windows Service Hosting