Project Overview: Barclays – Open Banking / PSD2 Compliance

Client:
Barclays Bank PLC
One of the largest multinational retail and investment banks headquartered in London, UK.

🗓️ Timeline:
November 2018 – May 2019

🌐 Domain:
Retail Banking
Regulatory Technology (RegTech)
Digital Banking APIs

📜 Regulatory Basis:
PSD2 (Revised Payment Services Directive) – EU directive mandating secure access to customer banking data by authorized third parties.
GDPR (General Data Protection Regulation) – Ensuring lawful handling of personal data and customer consent.

🎯 **Project Objective**
To develop and enable a secure, standards-compliant Open Banking API infrastructure that allows authorized Third-Party Providers (TPPs) to:

- Access account information (AIS)
- Initiate payments (PIS)
- Confirm fund availability (CAF)

All based on the customer's explicit consent, while ensuring data privacy and security under GDPR and meeting the compliance deadline imposed by EU regulators.

🧩 Key Drivers and Context

Driver	Description
🏛️ Regulatory Compliance	Required under EU PSD2 law by Q2 2019
🛡️ Data Security & Privacy	GDPR enforcement required enhanced consent capture and auditability
🤝 FinTech Collaboration	Open APIs enable partnerships with apps and platforms like budgeting tools, lenders, and payment services
💻 Digital Transformation	Shift from legacy systems to API-first architecture supporting secure, real-time banking

---

👨‍💼 Key Stakeholders

- Regulatory Compliance Team
- Retail Banking Product Managers
- Cybersecurity & InfoSec
- API Developers (Internal & Vendors)
- Legal & Data Protection Office
- Third Party Providers (TPPs)
- End Customers (Consent & UX)

🛠️ Scope of Work

Workstream	Description
API Enablement	Design, build, and expose RESTful APIs (AIS, PIS, CAF)
TPP Onboarding	Build developer portal, registration workflow, and digital certificates (eIDAS)
Consent Management	Implement OAuth2.0 + Strong Customer Authentication (SCA) for secure and explicit customer consent
Logging & Monitoring	Capture consent logs, access logs, API traffic reports, revocations, and usage analytics
Security Controls	Enforce encrypted payloads, penetration testing, and access token expiration policies
Customer Communication	Transparency through updated privacy notices, FAQs, and opt-in user interfaces

---

🔐 Compliance Standards Followed

Regulation	Details
PSD2	Mandated secure customer authentication and TPP access via APIs
RTS (Regulatory Technical Standards)	Defined by European Banking Authority for API security and SCA
GDPR	Consent logging, data minimization, and right to revoke access

---

💼 Tools & Technologies Used

Type	Tools
BA Tools	JIRA, Confluence, Visio, Excel RTM
API Dev	SwaggerHub, Postman, Apigee
Security	OAuth2.0, SAML, eIDAS certificates
Monitoring	Splunk, AppDynamics
Collaboration	SharePoint, MS Teams, Zoom

🏁 Outcome / Impact

- Successfully met PSD2 compliance deadline with zero fines.
- Enabled secure third-party access across millions of retail accounts.
- Strengthened customer trust via GDPR-compliant consent controls.
- Opened opportunities for collaboration with 50+ FinTech partners.
- Real-time monitoring of API usage helped detect early fraud attempts and reduced incident response time by 60%.

Phase 1: Project Initiation & Stakeholder Alignment

🎯 Goal: Define business objectives, clarify the regulatory scope, and align stakeholders across departments.

👋 **My Role as the Business Analyst in This Phase**

As the Business Analyst, I was responsible for initiating the project from the ground up by understanding the regulatory obligations under PSD2, defining a high-level scope, and aligning all relevant business and technology stakeholders.

This phase was crucial because PSD2 compliance was not only a technology transformation but also a legal and customer data privacy challenge. It required early, precise coordination across departments that normally operate in silos—like Compliance, Tech, Legal, and Product.

🧩 Step-by-Step Activities I Performed:

---

1. 📞 Conducted Stakeholder Kick-Off Meetings

---

2. 🧭 Defined the Project Scope and Regulatory Boundaries

Data Type	Exposure Type	Notes
Accounts	Read	Account type, number, and status
Balances	Read	Real-time balance at time of request
Transactions	Read	Past 90-day history (with customer consent)
Payment Initiation	Write	Initiate SEPA/UK Faster Payments with SCA validation

---

3. 📄 Created a RACI Matrix for Project Clarity

Activity	R	A	C	I
Define Data Scope	BA	Product Manager	Legal, DPO	Tech
Interpret PSD2 Requirements	Legal	Compliance Head	BA	Product
Define API Standards	API Architect	Tech Lead	BA	Legal
User Consent Flow	BA	UX Lead	DPO	Compliance

---

4. 🛠️ Set Up Tooling for Collaboration and Tracking

I established the foundational structure for our project documentation and task tracking:

🏁 **Outcome of This Phase**

By the end of Phase 1:

All key stakeholders were aligned on project goals and scope.

We had a clear, approved baseline of what APIs to build and what data to expose.

We avoided downstream ambiguity and ensured legal, privacy, and product teams were committed from the start.

Phase 2: Regulatory Requirements Elicitation

🎯 **Objective:**

To break down PSD2 and EBA Regulatory Technical Standards (RTS) into actionable business, functional, and non-functional requirements that could guide development, testing, and compliance.

👨‍💼 My Role as Business Analyst – What I Did

As the Business Analyst on the Barclays Open Banking API initiative, Phase 2 was the most regulatory-heavy stage. My job was to ensure that legal directives from the Revised Payment Services Directive (PSD2) and accompanying EBA RTS documentation were accurately interpreted and transformed into clear, traceable, and testable requirements.

✅ 1. Deep-Dive Collaboration with Legal & Compliance Teams

To start, I set up a series of working sessions with the Legal and Compliance teams, particularly those who had been tracking PSD2 and GDPR alignment.

- **Key Documents Referenced:**
  - PSD2 Directive 2015/2366/EU
  - EBA RTS on Strong Customer Authentication (SCA)
  - GDPR (specifically Article 6 – Lawfulness of Processing, and Article 7 – Conditions for Consent)

During these meetings, I acted as a translator—converting regulatory clauses into user-centric scenarios.

✅ 2. Created a Comprehensive Requirement Inventory

After analyzing the directives, I created a Requirement Inventory, categorizing each clause into:

- **Functional Requirements**
- **Non-Functional Requirements**
- **Compliance Rules**
- **Constraints**

The inventory included major features like:

Area	Requirement
TPP Identity	Validate eIDAS certificate using EU Trusted List
Consent Handling	Customer must explicitly grant and revoke TPP access
Session Expiry	Auto-expire sessions after X mins of inactivity
Access Limitation	Limit TPP access to 90 days unless re-authenticated
Data Logging	Maintain audit trails for all data access events

Each requirement was linked back to the clause number in PSD2 or RTS for traceability.

✅ 3. Identified Key Constraints Early

Worked closely with both the Data Privacy Officer and the Enterprise Security team to highlight must-follow boundaries, ensuring the API design wouldn't violate GDPR.
These constraints shaped the solution architecture discussions and were crucial for future UAT and InfoSec reviews.

✅ 4. Delivered the Business Requirements Document (BRD)

I authored the **BRD** in **Confluence**, with the following structure:

1. **Executive Summary** – Context of PSD2, goals of API enablement
2. **Scope** – Account types, channels, users, TPPs
3. **Regulatory Summary** – Key clauses impacting scope
4. **Detailed Business Requirements** – With unique IDs (e.g., REQ-PSD2-011)
5. **Assumptions & Constraints** – Based on GDPR, SCA
6. **Acceptance Criteria & KPIs**

✅ 5. Built the Regulatory Requirement Mapping Table (Clause → Feature)

One of the most critical artefacts I produced was the Regulatory Requirement Table, a matrix mapping:

- **PSD2 Clause / RTS Article**
- **→ Feature/Requirement**
- **→ Responsible System / Team**
- **→ Testable Condition**

PSD2 Clause	Requirement	Owner	Testable Criteria
Art. 66.3(a)	Validate TPP certificate	API Gateway	Must reject expired or invalid certs
RTS Art. 30	Consent logging	Data Logging Layer	Each access must have timestamp, TPP ID, user ID

This was **vital** for both audit trails and UAT planning later.

✅ 6. Started the RTM (Requirements Traceability Matrix)

To support downstream traceability, I created the first version of the **RTM (v1)** in Excel:

Req ID	Description	Source Clause	Jira Story ID	Status
REQ-001	TPP must be authenticated using eIDAS cert	PSD2 Art. 66	JIRA-OB-12	Draft
REQ-008	TPP access limited to 90 days	RTS Art. 36	JIRA-OB-17	Final

I tagged over 40+ initial requirements, categorizing them as:

- Frontend-facing (e.g., Consent UI)
- Middleware (e.g., session handling)
- API Gateway (e.g., cert verification)
- Audit Logging (e.g., trails, GDPR logs**)**

🔧 Tools Used

Tool	Purpose
Confluence	Drafted and managed the BRD, clause-to-feature tables
Excel	Maintained the RTM, Regulatory Matrix
JIRA	Created and tagged initial stories under the “PSD2 Compliance” Epic

Each requirement in Excel was hyperlinked to its corresponding JIRA ticket, ensuring traceability between documentation and backlog.

📌 Summary of What I Achieved in Phase 2

- Translated ambiguous legal texts into concrete, testable functional requirements
- Prevented future compliance risks by involving Legal, Security, and DPO early
- Delivered the BRD, Regulatory Matrix, and RTM v1—laying a foundation for Phase 3 (Functional Design)
- Ensured the project remained regulator-ready while aligning with internal tech constraints

---

Phase 3: As-Is / To-Be Process Modeling & Gap Analysis

🎯 **Objective:**
To conduct a side-by-side comparison of the existing (legacy) banking architecture and the proposed Open Banking architecture, with a focus on identifying functional, architectural, and regulatory compliance gaps.

👨‍💼 My Role -
This was a highly architecture-driven and process-mapping heavy phase. My main responsibility was to work closely with API Architects, Information Security, Legal, and Product stakeholders to model the transition from Barclays' internal data access system to the new PSD2-compliant API-driven framework.

🛠️ Step-by-Step Activities Performed

✅ **Step 1: Understand and Document the As-Is Architecture**
I began by thoroughly documenting the current customer account data access model in Barclays' legacy system. This was based on the internal Digital Banking Portal.

🔍 **Key As-Is Flow (Legacy Process):**
User logs into Barclays Online Banking.
Authenticates using credentials and 2FA (device + OTP).
Views accounts, balances, and transaction history.
All data stays within Barclays' internal ecosystem—no third-party involved.

📄 I captured this in BPMN using Visio to show user interaction layers, auth steps, and internal data fetch mechanisms.

✅ **Step 2: Define the To-Be Architecture (PSD2-Compliant Open Banking Flow)**
Next, I collaborated with the Open Banking Architecture team to visualize the future-state model that supports Account Information Services (AIS) and Payment Initiation Services (PIS) via Third-Party Providers (TPPs).

📈 I modelled this To-Be flow in BPMN 2.0 using Visio with the following swimlanes:

- Customer
- TPP
- Barclays Consent Service
- Barclays API Gateway
- Barclays Core Banking System

✅ **Step 3: Conducted the Gap Analysis**
After overlaying both models, facilitated 3 whiteboard workshops with security, legal, and API architects to perform a delta analysis.

🔍 Major Gaps Identified (12 total):

#	Area	Gap Identified	Risk Level	Suggested Fix
1	Consent Revocation	No central service to revoke TPP access	High	Build Consent Lifecycle Manager
2	Audit Logging	API access events not logged in legacy SIEM	High	Enable logging via Azure Sentinel

#	Area	Gap Identified	Risk Level	Suggested Fix
3	TPP Identity Verification	No certificate validation layer	Critical	Integrate eIDAS cert verification at API Gateway
4	Customer Notification	No user alert after third-party data pull	Medium	Add email/SMS push on every access
5	Time-bound Access	No mechanism to auto-expire access	High	Implement 90-day expiry cron logic
6	Legal Traceability	Data flows not tagged with regulation basis	Low	Add metadata tagging in logs

I compiled these into a Gap Analysis Report, categorized by:

- Functional gaps
- Technical gaps
- Compliance gaps

Each gap had a severity rating, impacted requirement ID (from RTM), and proposed solution with estimated effort.

✅ **Step 4: Delivered Artifacts**

📁 Artifact	Description	Tool
✅ As-Is Process Map	Modeled legacy banking interaction with data view-only flow	Visio (BPMN)
✅ To-Be Process Map	Modeled full PSD2-compliant TPP data access with consent and security layers	Visio (BPMN)
✅ Gap Analysis Report	12-point gap matrix with impact and remediation recommendations	Excel & Confluence
✅ Executive Summary Deck	For stakeholder review – focused on what changes, why it matters, and what's at risk	PowerPoint

---

📌 **Key Tools & Techniques Used**

Tool	Purpose
Visio	Process maps (BPMN 2.0) – As-Is and To-Be
Excel	Gap Matrix with sortable filters by severity
Confluence	Shared documentation space for review comments
Miro (optional)	Initial sketchboarding in workshops

🧠 **Impact in This Phase**

- Ensured stakeholder clarity on the architecture transformation.
- Helped API and Legal teams prioritize fixes before build phase.
- Provided visual artifacts that made compliance traceability possible in UAT.
- Prevented potential GDPR violations by highlighting the lack of revocation/audit features early.

Phase 4: Consent & Privacy Design (DPIA Phase)

🎯 **Goal:** Ensure that data-sharing with Third-Party Providers (TPPs) via Open Banking APIs complies fully with GDPR, using Privacy by Design principles and a Data Protection Impact Assessment (DPIA).

My Role -
At this stage, my role was to bridge Legal (specifically the Data Protection Officer), Security Architects, and Product Owners to ensure that all customer data flows — especially around consent — were designed legally, ethically, and transparently.

🧠 Step-by-Step Breakdown of My Work

✅ **1. Conducted DPIA Kick-Off with the DPO**
To begin, I arranged a working session with our Data Protection Officer (DPO) and stakeholders from the Security, Legal, and API Architecture teams. Aiming to:

- Understand the exact data categories being exposed.
- Clarify which data elements were considered Personally Identifiable Information (PII).
- Discuss any known privacy risks from similar Open Banking implementations.

✅ **2. Built the DPIA (Data Protection Impact Assessment)**
I led the creation of the DPIA, starting with a Data Inventory and Risk Mapping, following ICO (UK Information Commissioner's Office) guidelines.

🔐 Risk Identification:
For each data category, I identified potential GDPR risks.

✅ Mitigation Strategies Proposed:
Implement token-based access using OAuth 2.0 (short expiry tokens, refresh logic).
Maintain Consent Logs with timestamps, scope, and TPP ID.
Build Real-Time Revocation endpoints (immediate effect).
Implement Auto-expiry (90 days) with pre-notification reminders.

Then filled all of this into our Barclays DPIA Template, which included:

Section	Description
Data Types Processed	Financial metadata (not raw identity)
Processing Purpose	Regulatory (PSD2), Customer-Initiated
Legal Basis	Article 6(1)(c) – Legal obligation
Risks Identified	Unauthorized access, Profiling, Token leakage
Mitigation Controls	OAuth2.0, SCA, Token Scoping, Consent UI

I then uploaded the final DPIA for legal approval via Confluence and sent it through the standard risk review process.

✅ **3. Designed Consent & Privacy UX Flows**
Once the DPIA risks and requirements were documented, I worked directly with Product Designers and API Teams to design and document three key flows:

🔁 **A. Consent Granting Flow**
📌 Purpose: When a user authorizes a TPP to access data

📐 **Tools Used:**
Designed in Visio using BPMN 2.0 + UI screen annotations

🔄 **B. Consent Revocation Flow**
📌 Purpose: When a user wants to revoke previously granted access

📐 Tools Used:
BPMN in Visio + API response mapping (token expiry confirmation)

⏱️ **C. Session Expiry & Notifications**
📌 Purpose: Handle session management and consent expiration

📐 Tools Used:
Sequence diagrams (Miro)
Notification logic reviewed with customer experience and comms team

---

📂 Deliverables I Produced

📄 **Artifact**	**Description**
✅ DPIA Document (Word + Confluence)	Fully filled DPIA with data types, processing purpose, risks, and controls. Mapped to GDPR Articles. Reviewed by Legal & Infosec.
✅ Consent Flow Diagrams (Visio BPMN)	Included three core flows: Consent Grant, Revocation, Session Expiry. Annotated with API callouts and UI events.
✅ Consent & Security Audit Trail Specification	Designed log schema:
— Consent ID	
— TPP ID	
— Timestamp	
— Action (grant/revoke/expire)	
— IP address	
→ Stored in GDPR-compliant log service. Shared with DevSecOps for implementation.	

💡 Key GDPR Articles Considered

Article	**Relevance**
Art. 6(1)(c)	Legal basis for processing = PSD2 obligation
Art. 7	Conditions for valid consent
Art. 25	Privacy by Design – token scoping, access controls
Art. 30	Record of processing activities (covered in DPIA)
Art. 32	Security of processing – encryption, logging
Art. 5	Principles: Data minimization, purpose limitation

Phase 5: Functional Requirements Breakdown & API Design

🎯 Objective: Define clear, regulatory-aligned functional and technical requirements for Open Banking APIs, ensuring compliance with PSD2 and high performance for external TPPs.

👨‍💼 My Role as the Business Analyst

In this phase, I acted as the bridge between business regulation, compliance requirements, and technical API implementation. My job was to break down the business objectives of PSD2 into practical, secure API interfaces, while also defining and refining each endpoint, field, and rule the bank would expose to licensed Third Party Providers (TPPs).

🔹 **Step 1: Conducted API Requirement Workshops**

I began by organizing a series of working sessions with:

- API Architects
- Open Banking Product Owner
- InfoSec & Legal Teams
- Back-end Integration Team
- Compliance Officers

These sessions were structured around each API endpoint required under PSD2.
I facilitated these workshops using JIRA Epics as discussion anchors, ensuring each endpoint had a clear:

- Business justification
- Regulatory clause
- Functional scope

🔹 **Step 2: Defined API Endpoint Structures & Fields**

Using Swagger UI (mock environment) and Confluence, I led the effort to define and finalize:

✅ /accounts Endpoint
✅ /balances Endpoint
✅ /transactions Endpoint
✅ /payments Endpoint

Step 3: Finalized Security & Error Specifications

I worked closely with the API security team to ensure:

- eIDAS QWAC validation via mutual TLS
- OAuth2 + PSD2-compliant consent tokens
- Proper handling of 401 Unauthorized, 403 Forbidden, 429 Too Many Requests, and 500 Internal Server Error

These were documented in a Confluence-based API error matrix, cross-linked with Swagger-generated mocks.

🔧 **Step 4: Non-Functional Requirements (NFRs)**

As BA, I was also responsible for non-functional specifications to ensure performance, reliability, and auditability.

Key NFRs I Defined:

Metric	Requirement
Uptime SLA	≥ 99.5% monthly availability
Latency	<300 ms for 95% of API calls
Throughput	Handle 100 requests/sec sustained load
Timeout	10 seconds hard timeout for each API call
Audit Logging	Every call logged with timestamp, TPP ID, consent ID
Rate Limiting	Per TPP basis (dynamic based on risk)

Each NFR was mapped to a specific risk mitigation, especially to align with GDPR and EBA guidelines.

📁 **Deliverables I Produced**

Deliverable	Description	Tool Used
✅ API Specification Tables	Detailed tables listing fields, descriptions, data types, formats	Excel + Confluence
✅ Endpoint-Level User Stories	JIRA Stories for each API method with ACs	JIRA
✅ Field Mapping Sheets	Mapped backend core fields (e.g., CBS transaction objects) to Open Banking fields	Excel
✅ Security Header Documentation	Defined required headers (x-request-id, Authorization, Date, etc.)	Confluence
✅ Swagger API Contracts (Mocked)	Worked with developers to generate and test Swagger contracts	SwaggerHub
✅ NFR Matrix	Tabular view of latency, uptime, throttling, audit logging	Excel + Shared in Confluence

---

Summary:

- Acted as the functional owner of the Open Banking API suite from the BA side.
- Translated regulatory text into endpoint-level, testable specifications.
- Worked daily with technical architects to align business goals with JSON specs and OAuth flows.
- Provided a single source of truth via Confluence for all API behavior and SLA expectations.
- Ensured all requirements were traceable to PSD2 clauses via RTM (v2.0, extended).

Phase 6: Testing & UAT Execution

🎯 Goal: Ensure the entire API ecosystem works end-to-end—from TPP registration and consent flow to secure data access—meeting both functional correctness and regulatory compliance before go-live.

As the Business Analyst, I was directly responsible for orchestrating the UAT lifecycle for the Open Banking initiative. This was a critical go/no-go stage, especially with regulatory oversight from the FCA and strict conformance to PSD2 standards.

🔍 Step 1: UAT Planning & Environment Setup
I began by collaborating with:

- The QA team (for test data provisioning)
- The API architects (to confirm endpoints and payloads in the staging environment)
- The Legal and DPO teams (to ensure all test cases reflect real consent and privacy flows)
- I validated the following were in place before testing began:
- OAuth2 tokens from a sandbox TPP
- Valid eIDAS certificates for TPP simulation
- Test accounts with varied transaction and balance histories
- Expired, revoked, and reauthorized consent tokens for negative path testing
- We used JIRA to manage the test cycle and Excel to document the UAT test cases.

✅ **Step 2: Writing UAT Test Cases (70+ Written)**
I created over 70 UAT test cases, mapped directly to the RTM and BRD. These included:

- Positive Flow Scenarios
- Negative Test Scenarios

Each test case included:

- Pre-conditions (token, TPP status, account setup)
- Steps
- Expected HTTP response code
- Payload validation logic
- GDPR compliance notes

🐞 **Step 3: Defect Logging & Triage via JIRA**
Over the testing window, 42 defects were logged in JIRA. I personally led the daily triage calls with QA, development, product owners, and security architects.
Defect Breakdown:
Every issue was assigned a priority (P0–P3) and linked to the corresponding Epic or Story in JIRA.

📜 **Step 4: Final UAT Report & Sign-Off**
I prepared a UAT Summary Report covering:

- Test Execution Status (Passed/Failed/In Progress)
- Defect Matrix by Severity
- Risks and Blockers
- Compliance Coverage Matrix (mapping test cases to PSD2 clauses)

Conducted the final UAT sign-off meeting with:

- Product Manager (for feature validation)
- Head of Compliance / DPO (to verify privacy/consent behavior)
- Lead Architect (for performance & stability sign-off)
- We received formal sign-off to move forward with go-live, having fixed all P0–P1 defects, and logged known non-critical items for post-release sprints.

📄 Deliverables Produced

Deliverable	Description
✅ UAT Test Case Suite (Excel)	70+ scenarios, all mapped to RTM & BRD
✅ UAT Execution Log	Pass/fail status, tester name, date, and environment
✅ JIRA Defect Tracker	Linked to stories and test cases
✅ UAT Summary Report	Sent to all stakeholders

Deliverable	Description
✅ Consent & Privacy Test Report	Used for GDPR audit trail

Phase 7: Go-Live & Post-Implementation Support

🎯 **Goal:** Ensure a stable go-live of production APIs, validate initial third-party integrations, and capture early feedback for continuous improvement.

My Role
As the Business Analyst, I was deeply involved in monitoring, validating, and supporting the go-live of our Open Banking APIs. My responsibilities went far beyond documentation — I was embedded in the cross-functional launch team, working alongside API developers, DevOps, Legal, and our external fintech partners.

🔧 **1. Pre-Go-Live: Final API Validation with Fintechs**
Before launch, supported final sandbox testing sessions with our early-access third-party providers (TPPs) — notably:

- Yolt
- TrueLayer
- Plaid (US-based, testing EEA readiness)

I conducted test case walkthroughs with each TPP's engineering team, ensured consent flows worked with their apps, and verified correct handling of:

- OAuth2 tokens
- Consent expiration flags
- Error handling (401, 403, 429)

📊 2**. Go-Live Monitoring: Key Metrics & System Health**
During the first 2 weeks of production, actively monitored the following KPIs via our real-time dashboards (Splunk + API Gateway + Azure Monitor):

Metric	**Monitoring Tool**	**Result**
API Call Volume	Splunk Dashboards	192,000+ API hits (Week 1)
Consent Revocations	Internal consent audit log	~230 revocations
Uptime %	Azure Monitor SLA dashboard	99.6% uptime
API Response Latency	Splunk traces	<250ms avg
Success Rate	Internal log filters	97.6% API success (non-5xx/4xx)

I created daily health-check summaries on Confluence to update all stakeholders and flagged any anomalies or unexpected spikes.

💬 **3. Feedback Collection from Fintechs**
I personally reached out to our partner TPPs after their first week of production usage to collect qualitative feedback.
I captured this in the Fintech Partner Feedback Summary.

✅ 4. Final Deliverables Created

Deliverable	Description
🚀 Go-Live Checklist	Included 24 readiness criteria: API versioning, SLA compliance, error response formats, DPO sign-off
📈 Launch Report	Contained usage stats, latency benchmarks, defect logs, fintech feedback
📋 Fintech Feedback Summary	Consolidated direct input from 5+ TPPs
📂 CR Backlog Spreadsheet	Prioritized list of post-Go-Live improvements with business rationale

All artifacts were stored in Confluence → /OpenBanking/Go-Live-Support/2025 and attached to the Sprint Review.

👥 Cross-Team Collaboration
During this phase, I actively engaged with:
DevOps & Platform Engineers → for real-time alerting and issue resolution
Product Owner → to log CRs and prioritize enhancements
Legal/DPO → for compliance checks on data sharing and audit logs
External TPPs → for direct developer-level support

🎯 Key Contributions as a BA in This Phase-

- Ensured regulatory functionality performed under real-world usage.
- Converted fintech and platform feedback into prioritized backlog items.
- Provided real-time visibility of post-Go-Live KPIs to leadership.
- Enabled smooth onboarding and early adoption by 37 TPPs.
- Verified that GDPR and PSD2 constraints remained intact post-launch

Final Phase – Business Analyst Deliverables Summary

✅ **1. Business Requirements Document (BRD)**
📍 Tool Used: Confluence
I authored the BRD in Confluence to ensure traceability, version control, and collaborative access across teams.
The BRD covered:

- Functional requirements for all mandated endpoints (/accounts, /balances, /transactions, /payments)
- Regulatory logic extracted from PSD2 RTS and mapped to consent and security flows
- User journeys for:
  - TPP onboarding
  - Consent grant/revoke
  - API invocation lifecycle

The BRD was reviewed and approved by Product Owners, Legal, and Technology leads, and became the anchor document for downstream design and testing teams.

---

2. Requirements Traceability Matrix (RTM)

📍 Tool Used: Excel

This was one of the most critical compliance artifacts. I built the RTM in Excel, mapping:

- 40+ regulatory clauses from PSD2 and EBA RTS
- Corresponding features (e.g., "90-day re-authentication", "eIDAS cert validation")
- Linked JIRA Story IDs and BRD sections

It allowed auditors and internal risk teams to validate that every legal requirement was fully implemented and tested.

---

3. Data Protection Impact Assessment (DPIA)

📍 Tool Used: Word (Standard GDPR DPIA Template)

Collaborating with the Data Protection Officer, I completed a DPIA to evaluate potential privacy risks. Sections I filled included:

- Data flow mapping: IBAN, account balances, and transaction data
- Risk identification: Exposure of financial data to third parties
- Mitigation strategies:
    - Token-based access instead of credentials
    - Time-bound consent and audit trails
    - Encryption & logging

The DPIA was signed off by DPO and Legal and submitted as part of the PSD2 compliance audit.

---

4. Consent Lifecycle Flow Diagrams

📍 Tool Used: MS Visio (BPMN Notation)

I created BPMN-style diagrams to capture:

- Consent Granting Flow: Customer initiates consent via TPP → Consent forwarded to Barclays → eIDAS cert validation → Data access granted
- Consent Revocation Flow: Revocation via TPP or customer → Consent revoked → Data access blocked instantly
- Expiry scenarios: Automatic revocation after 90 days → User notification triggers

These diagrams were critical in driving alignment between API Engineering, Security, and Compliance.

✅ **5. As-Is and To-Be Architecture Maps**

📍 Tool Used: MS Visio

I modeled two key states:

- As-Is: Customers manually log into Barclays Online Banking to view data; no external API access.
- To-Be: Authorized TPPs access APIs post-consent; all activity is monitored and auditable.

This comparison fed into the Gap Analysis workstream and clarified exactly what needed to change at the system architecture level.

---

6. UAT Plan & Defect Log

📍 Tool Used: Excel & JIRA

I created a structured UAT plan including:

- 70+ test cases (positive and negative)
- Traceability to BRD and RTM
- Functional test scenarios like:
    - Token validation
    - Consent expiry
    - Revocation triggers

Using JIRA, I managed defect triage:

- 42 defects logged
- 12 were logic-related
- 9 were performance-related (timeouts, throttling)

Weekly UAT triage calls ensured timely resolution before go-live.

✅ **7. Gap Analysis Report**

📍 Tool Used: Excel

I documented 12 key architectural and process gaps, uncovered by comparing the "To-Be" Open Banking model with Barclays' legacy system.

Example gaps:

- No audit trail on revocations in legacy
- Consent expiry logic missing from session manager
- Lack of auto re-authentication trigger at 90 days

Each gap was logged, assigned to a technical owner, and tracked to closure in our delivery backlog.

🧩 Business Analyst Skills Demonstrated

This project demanded a broad and deep BA skillset:

Skill	**Where It Was Used**
Regulatory Analysis	PSD2 & GDPR clause breakdown in BRD & RTM
Process Modeling	BPMN flows in Visio for consent lifecycle
Data Privacy Design	DPIA creation with DPO
Stakeholder Communication	Alignment across Legal, Product, API Engineering
UAT Planning	Test case creation, defect triage
Documentation Excellence	RTM, BRD, Consent Flowcharts, DPIA
Tool Mastery	JIRA, Confluence, Excel, Visio, Word

🎯 **Project Impact**

- Barclays became one of the top 3 UK banks to go live with full PSD2 API compliance — well ahead of the regulatory deadline.
- We enabled secure, GDPR-compliant financial data sharing via Barclays Developer Portal.
- Fintech partners like Yolt and TrueLayer onboarded seamlessly, thanks to our clean consent flows and endpoint clarity.
- Customers now have greater control and transparency over who accesses their financial data.