# **📗 Onboarding workflow**

## **1. Problem Statement**

The current **client onboarding workflow** is **manual, static, and Excel-driven**.  
 Each onboarding case requires multiple spreadsheet updates, email confirmations, and manual task tracking.

It exposes below limitations:

* Limited visibility — only one team member at a time tracks client progress.
* Little automation - manual work load across clients and teams.

## **1.2 Existing POC Summary**

A **simplified proof of concept (POC)** has been created for **single-user onboarding** of a **new client using mocked up file and status**.  
It uses:

* **Streamlit** as front end for user engagement and status rendering.
* **AWS Bedrock API** for AI chat conversation and workflow transition.
* **Local file storage** for client and workflow definitions.

### **POC Use Case**

* A user selects a **mock client profile** from file storage.
* The app loads a predefined mocked workflow and task lists (e.g., collect KYC and etcs) through md files and coded task transitions using prompts.
* The user updates each task manually through chat which triggers task updates and next steps.

### **Observations and Limitations**

The existing Proof of Concept (POC) has several limitations:

* **Mocked Scenarios:** The current user cases are simulated and may not accurately reflect real-world business scenarios.
* **Workflow:** The workflow itself needs refinement.

**Technical Limitations (Streamlit):**

* **Reload Delays:** Streamlit re-runs the entire Python script with every click or input, leading to noticeable delays.
* **State Management:** Application state is stored in the cache and can easily reset without explicit handling, making it challenging to maintain continuity.
* **Multi-User Support:** The POC lacks native real-time multi-user support and shared session state, limiting collaborative use.
* **UI Customization:** There is no native support for popular front-end libraries like React or JavaScript, which restricts advanced UI customization.

Despite these limitations, the POC successfully demonstrates the concept of guided onboarding and highlights the potential for AI-assisted workflows. Future improvements should concentrate on:

1. Refining the POC's business use cases.
2. Implementing robust workflow orchestration.
3. Developing support for multi-user roles and persistent client/task tracking.
4. Enhancing the overall User Interface/User Experience (UI/UX).

## **2. Goal**

The goal of this project is to **build a standardized and extensible onboarding workflow POC** that improves client-specific customization.

### **Objectives**

1. **Standardize a simplified workflow template** with predefined tasks and statuses (P0).
2. Enable **dynamic task progression** — completing one task moves the workflow forward.
3. Support **multiple user roles** with different levels of interaction (read, edit, admin P1).
4. Allow **admins to edit workflows and tasks** through configuration rather than code.

## **3. User Stories**

### **3.1 Read-Only User**

*“As a read-only user, I want to view the client profile and current onboarding status, so I can track where the client stands in the onboarding process.”*

* Can see all workflow stages and task completion status.
* Cannot modify any data or mark tasks complete.
* Typical roles:TBD

### **3.2 Edit-Role User**

*“As an edit-role user, I want to update onboarding tasks by entering required information, so the workflow can automatically move to the next stage once completed.”*

* Can mark tasks as complete or update data fields.
* The system updates client status and task state dynamically.
* Typical roles: TBD

### **3.3 Admin User**

*“As an admin, I want to modify and update workflow templates, tasks, and client types, so that the system can adapt to different onboarding requirements.”*

* Can add, update, remove workflow stages and tasks as template for SMA and fund user.
* Can add, update, remove workflow for particular user.
* Responsible for baseline workflow governance.

## **4. Functional Requirements**

| **ID** | **Function** | **Description** |
| --- | --- | --- |
| F1 | Client Profile/Task/Workflow View (Existing) | Display key client fields (name, type, onboarding status, assigned tasks). |
| F2 | Task Progress(P0) | Allow edit-role users to update, complete, or reopen tasks. |
| F3 | Workflow Progression (P0) | Automatically move workflow to the next stage once all tasks in a stage are complete. |
| F4 | Role-Based Access (P1) | Enforce read-only, edit, and admin permissions. |
| F5 | Workflow Template Management (P1) | Admins can define templates (baseline + customized tasks). |
| F6 | Audit Trail & History (P2) | Log who completed which task and when (optional for later phase). |

## **5. Workflow Examples**

### **Example 1: New Institutional Client Onboarding**

****Workflow: Institutional Client Onboarding

Client: ABC Insurance Ltd.

Status: In Progress

Stage 1 – Client Information

▢ Collect Client KYC

▢ Verify Corporate Docs

▢ Assign Relationship Manager

Stage 2 – Account Setup

▢ Define Investment Vehicle (Fund or SMA)

▢ Create Account in System

▢ Upload Initial Funding Docs

Stage 3 – Compliance Review

▢ Check AML and Sanctions

▢ Verify Tax Residency (Forms W8/W9)

▢ Obtain Approval from Compliance Officer

Stage 4 – Finalization

▢ Confirm Activation Date

▢ Send Welcome Package

▢ Update Status → Onboarded

When all tasks in a stage are marked complete, the workflow transitions to the next stage.  
 If a compliance task fails, the system can revert to the previous stage for corrections.

### **Example 2: SMA Client Onboarding**

****Workflow: SMA Client Onboarding

Client: UV Insurance SMA

Status: Pending

Stage 1 – Initial Setup

▢ Collect Client Contact Details

▢ Define Portfolio Guidelines

Stage 2 – Custodian Integration

▢ Register Custodian Account

▢ Upload Holding File Template

▢ Assign Investment Manager

Stage 3 – Activation

▢ Verify All Data Inputs

▢ Run Test Trade Cycle

▢ Update Status → Live



## **6. Future Enhancements**

* Multi-user concurrent access (shared workflow dashboard).
* Persistent backend (database instead of files).
* Improved UX in Streamlit (accordion views, progress bars, audit logs).
* AI assistant via Bedrock for task guidance and text summarization.

## **7. Next Steps & Implementation Plan**

| **Phase** | **Description** | **Deliverable** |
| --- | --- | --- |
| 1 | Define standardized onboarding workflow template and baseline tasks. | Template YAML / CSV |
| 2 | Implement Streamlit UI with role-based access (read, edit, admin). | Streamlit v1 Prototype |
| 3 | Integrate AWS Bedrock for contextual help / task suggestions. | Bedrock API Integration |
| 4 | Conduct testing with mock clients and sample workflows. | UAT Report |
| 5 | Document learnings and prepare design for persistent backend (Phase 2). | Design Spec |

## **8. Summary**

This PRD defines an **AI-ready onboarding workflow system** aimed at replacing Excel-based manual processes.  
 It begins with a **Streamlit + AWS Bedrock POC** that supports multiple roles, standardized tasks, and dynamic progression.  
 The next steps focus on establishing a solid workflow template foundation and refining the user experience before scaling to a multi-user backend architecture.