Requirement Doc - Backbone to React (ES6 Compatible) using mPower API

Applicable Release: E2 only

1. Product Development Use-cases:
2. Upgrade Use-cases: &

1. Objective

Migrate all Form-level Backbone JS (stored in the Form metadata) to ES6-based React, leveraging the latest mPower APIs and Business Rules (BR) Guide to ensure full compatibility with E2.

2. Background

- **Legacy:** In E1 and earlier, developers embed custom Form logic in Backbone JS, in Form metadata.
- **Upgrade Impact:** After upgrading to E2, this legacy code may fail or behave unpredictably.
- **API Update:** E2 have updated mPower APIs; these needs to be used in place of the Backbone JS code.

3. Scope

- **In-Scope:**
    - All Forms contain Backbone JS.
    - Conversion of that code to React (ES6) syntax.
    - Integration with the latest mPower API methods.
    - Commenting out legacy JS to allow safe rollback.
    - Use BR API in place of JS logics wherever applicable

4. Conversion Scenarios

Scenario	Action
A. No existing mPower API calls in the existing backbone JS	1. Rewrite Backbone JS logic as React (ES6) compatible format. 2. Replace JSs with appropriate BR and mPower API calls.
B. Contains legacy mPower API calls in the existing backbone JS	1. Refactor existing API calls to match the latest BR & mPower API. 2. Rewrite surrounding logic in ES6, using React patterns.

5. Current Team Process

How the Product Developers currently performs conversions

1. **Discover:**
    1. Navigate to Module → Forms → select target Form.
    2. Extract Backbone JS from metadata or File System.
2. **Analyze & Copy:**
    1. Create a working copy of the original JS.
    2. Annotate key logic paths and API interactions.
3. **Convert:**
    1. Translate overall logic, control flow and data handling into React ES6 components.
    2. Replace or update API invocations using the mPower API Guide.
    3. Apply BRs for any inline business logic per the BR Guide.
    4. Comment out original Backbone blocks for easy rollback.
4. **Validate & Deploy:**
    1. Lint and build (ESLint/prettier).
    2. Functional test within the E2 environment.
    3. Paste converted code back into Form metadata and test functionality.

How the CSS/Upgrade/Partner Developers currently performs conversions

1. **Discover:**
    1. Navigate to Module → Forms → select target Form.
    2. Extract Backbone JS files (<FORMNAME>_pre.js, FORM_NAME>.js and <FORM_NAME>_post.js from metadata or File System.

2. **Analyze & Copy:**
    1. Create a working copy of the original JS files.
    2. Merge all 3 JS files into a single file.
    3. Annotate key logic paths and API interactions.
3. **Convert, Validate & Deploy:**
    1. This is similar to the Product Developer does.

6. Admin-UX Conversion Agent Expectations

6.1 Content Conversion Workflow

- **JS Input:**
    - Provide a field for developers to paste existing Backbone JS or attach the file, from Form metadata or File System.
    - Option to provide multiple files (Pre & Post along with FORM_NAME>.js)
- **Conversion Plan Generation:** Automatically outline a step-by-step plan that includes:
    1. Merge the files into a single file and analyze.
    2. ES6 module structure (imports/exports, components, etc..).
    3. mPower APIs for corresponding JS logic. Generate and consider the syntax grammar for mPower APIs for JS logic. Having syntax grammar would avoid syntactical validation and it would be first time right.
    4. BRs for corresponding JS logic following BR guide.
    5. Placeholder points for any custom UI logic or business rules, based on Developer's additional asks for Business logic.
- **Feedback & Customization:** Allow developers to review and edit the plan or API selections; support inline comments.
- **Additional Logic Integration:** Let developers specify extra UI elements (e.g. buttons or handlers) or additional business logic so the plan and final code include these.
- **Code Generation:** On approval, produce React (ES6) code embedding latest mPower API calls BR API calls and any custom logic. This conversion should happen on the final merged file.
- **Copy & Deploy:** Enable easy copying (or download the converted file) of generated code for pasting into file system and testing.

6.2 Editor Use Case

- **FORM Discovery:** UI to browse Modules → Module → Objects → Forms → select target Form.
- **Metadata Retrieval:** Read and display current Backbone FORM JS from metadata or File System.
- **Guided Conversion:** Invoke the Content Conversion Workflow with review and approval controls.

6.3 Testing & Validation:

- Verify syntactical correctness (lint + build).

- Check feasibility of Execute functional tests on targeted Form to confirm if Form is working as per expected behaviour.

7. Reference

1. **mPower API Guide** – latest method signatures and examples in Platform mPower APIs - Technical Publications - Confluence
2. **Business Rules (BR) Guide** – syntax and patterns for inline rules
3. Existing OOTB JS code conversion to React equivalent - M7 Platform - Confluence
4. JS changs - R&D Operations - Confluence
5. ChatGPT - Uses in Development - Team Apps - Confluence