Upgrade AI - JS to BR (Requirement Document)

Great — let's now expand the user story into a more **detailed and enriched version** to cover potential edge cases, alternate flows, and UI/UX clarity that will better support the development and QA teams.

User Story: Enhanced JS to BR Conversion – AgenticAI Upgrade Module

Feature Title:

AgenticAI Upgrade Module – JavaScript to Business Rule (JS to BR) Conversion

User Persona:

Primary User: Developer/Automation Engineer familiar with MetricStream Forms
Secondary User: QA Engineer verifying BR correctness and Unit Test coverage

---

User Story (Expanded):

In the JS to BR Conversion , Developer will Upload JS file to convert the JS code into Business Rules.

1. Users will login to the AgengticAI platform.
2. Through the Left-pane, User will be able to hover to the Upgrade section in which he will be able to see three options:
   1. JS to BR
   2. BackboneJS to ES6
   3. PL-SQL to Emery
3. User will be able to selection the JS to BR option to covert JS files to respectove BR's wherein users will be able to see
   1. Instruction Pane
   2. Project Plan Pane (part of the Execution Pane)
   3. Results which include Generated BR's (part of the Execution Pane)
4. Users can use the attach button from Instruction Pane to select the JS files which needs to be uploaded . Users can upload the root JS file for which the BR's needs to be generated along with dependent JS files necessary to generate the BR's.
5. The LLM will parse the JS files to search for the dependencies that the files contain. If necessary files are not uploaded based on the parsed results, LLM should also prompt the users to upload all the necessary files which were not uploaded earlier.
6. Once users confirm that all the files are uploaded, LLM will prompt for the plan generation workflow which needs to happen before generating BR's.
7. Upon confirmation of the users , LLM will generate the plan which would include steps through which the functions and objects in the root file will get converted to respective BR's in the Execution Pane.
8. There will be Modify and Execute buttons displayed in the generated plan.
9. If the generated plan suffices the requirements of the users to generate BR's , users can click on the Modify button

which will take users to Instruction pane from where users will be able to give prompts to make necessary changes to modify the plan.

10. If the generated plan suffices the requirements of the users to generate BR's , users can click on execute button to start the BR generation process.
11. Once the Plan is generated successfully , the BR generation gets started upon confirmation of users.
12. In the BR generation workflow , the JS file gets parsed for objects, functions and dependencies on the other uploaded files to convert related JS code into respective BR's.
13. Once the BR's are generated the results of the BR's are shown in the execution pane , just below the plan.
14. The table for the generated BR's is expandable which shows 4 columns as below:
    Field Name: Contains name of the field for which the BR has been generated.
    BR Category: Contains the category of the field for which the BR has been generated.
    Generated BR: Contains the generated BR
    Unit Test : Contains the number of unit test passed/failed executed in code sandbox.
15. Each row in the table is expandable which will give user the complete BR that is generated along with the related JS code from which the BR has been generated.
16. If the unit tests are failed or the generated results are incomplete, Users can modify the results by giving the LLM prompts to do any specific changes in BR's or to run unit tests against specific BR's using instruction pane.
17. Once the BR's are generated Users can copy the BR's and use it in the Forms in the target instance which will be a manual process that users need to do in phase-1.
18. The Upgrade UI will create updated versions of the uploaded files in which the JS code which is used to generate BR's is commented and these files will be downloadable for users.
19. Users can access previous conversations using History in the Left Pane using which they will able to revisit the previous conversations they have had with conversion agents.

Preconditions:

- The user must be logged in.
- The user has access to the "Upgrade" section.
- The required JS files (main and dependencies) are available locally.

Core Functional Flow:

1. Navigation and Access:

- The user logs in to AgenticAI and clicks **Upgrade** in the left navigation.
- Three options appear:
    1. JS to BR
    2. BackboneJS to ES6
    3. PL/SQL to Emery
- The user selects **JS to BR**.

2. Instruction Pane UI:

- The Instruction Pane includes:
  - Guidelines (e.g., “Attach all necessary files before conversion”)
  - **Attach** button to upload JS files
  - Optional command input to interact with the LLM for clarifications

3. File Upload and Dependency Validation:

- The user uploads:
  - A root JS file (the primary file for BR generation)
  - Supporting files (e.g., utility modules or shared constants)
- The system parses the uploaded files for:
  - `import` or `require` statements
  - Function or constant usage from missing files
- If missing, LLM displays a message like:

  "Module './MS_GRC_COMMON.js' not found. Please upload this dependency."
- User is blocked from proceeding until all required files are uploaded.

4. Plan Generation:

- Upon successful upload, the user is prompted to generate a plan.
- The plan contains:
  - Sequential steps (e.g., parse → extract function logic → identify BR conversion targets)
  - Estimated conversion time
  - Highlighted JS functions to be translated
- Shown in the **Project Plan Pane** under Execution Pane
- User can:
  - **Modify** the plan via conversational input
  - **Execute** the plan to proceed

5. Business Rule (BR) Generation:

- LLM:
  - Extracts business logic from uploaded files
  - Uses domain-specific knowledge to map JS constructs (e.g., field validation) to BR format
- Generated BRs appear as a table:

  Field Name BR Category Generated BR Unit Tests

- User clicks to **expand** any row to view:
    - Extracted JS snippet
    - Converted BR rule
    - All test cases executed with results (e.g., “3/3 Passed”)

6. Unit Testing and Troubleshooting:

- For every BR, test cases are automatically generated and run in a sandbox.
- If a BR fails:
    - LLM suggests fixes
    - User can re-run unit tests or refine BR logic through prompts
- Failed test indicators (e.g., red □ or “2/3 Passed”) are shown per row

7. Output Management:

- Users can:
    - **Copy** individual BRs to clipboard
    - **Download** modified JS files (commented-out converted logic)
- Manual intervention is required to paste BRs into target forms (Phase-1)

8. Conversation & History:

- The left pane shows **History** (conversion sessions).
- Each session is timestamped and includes:
    - Uploaded files
    - Generated BRs
    - Conversion plan
    - LLM prompts and responses
- Useful for version tracking and audits

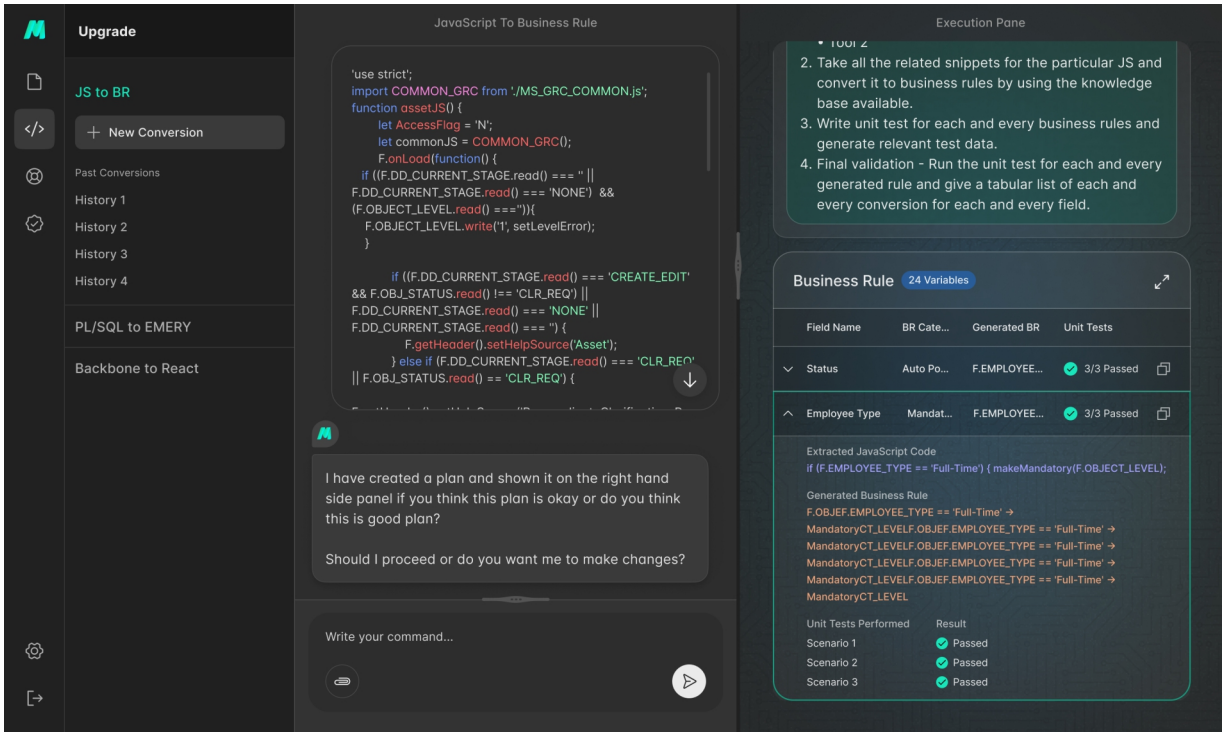
Alternate Flows & Edge Cases:

Case	Description	System Response
Missing Dependency	Required module not uploaded	Prompt user and list missing files
Invalid JS	JS file contains syntax error	Alert user and identify the error
No Convertible Logic	No functions map to BRs	Inform user and skip BR generation
Plan Rejected	User doesn't accept generated plan	Allow modification or start over
Test Failure	BR fails unit tests	Show debug info; allow retest

---

UI/UX Considerations:

- Progress bar during plan or BR generation
- Status indicators (e.g., spinner, success checkmark)
- Tooltip descriptions for each BR column
- "Copy All" button to export all BRs at once
- Download button with filename auto-versioning (`converted_asset_v1.js`)



Would you like to now move forward with this version to generate the **Functional Requirement Document (FRD)**?