Rule Check

Introduction / Purpose:

The Rule Check system is designed to validate reimbursement requests and enforce business policies in a consistent, automated, and auditable way. The system provides employees with transparency, policy makers with tools to manage rules, and auditors with reliable access to compliance data. Its goal is to ensure company funds are used appropriately while maintaining accountability.

Table of Contents:

Actors / Roles:

- **Employee:** Submits reimbursement requests for review.
- **Policy Maker:** Creates, edits, and deletes policies and rules that govern reimbursement decisions.
- **Auditor:** Reviews logs and decisions to ensure compliance and investigate activity.

Glossary:

- **Policy:** A set of business rules that define acceptable and unacceptable expense types.
- **Rule:** A specific condition or restriction attached to a policy.
- **Action Submission:** A request submitted by a user (e.g., reimbursement request).
- **Decision:** The system's determination to accept or deny an action submission, with an explanation.
- **Audit Log:** A record of all submissions, decisions, and system actions, used for compliance review.

User Cases:

1. **As an Employee**, I want my reimbursement submissions to be automatically checked against company policies, so only valid business expenses are reimbursed.
    a. **Outcome:** The Employee's submission is automatically validated, and non-business reimbursements are blocked.
    b. **Requirements:** The system must be able to evaluate a user request for reimbursement, compare their expense to the correct policy and its rules, return the decision and the generated natural language explanation of the decision, and log the transaction.
    c. **Inputs:**
        i. User ID.
        ii. Action type.
        iii. Expense detail.
        iv. The respective policy and its rules.
    d. **Outputs:**

        i. To the user: The decision and the natural language explanation of the decision.
        ii. To the system: The audit logs.
    e. **Constraints:**
        i. The system must always use the active policy (not outdated drafts).
        ii. The decision and audit log must be immutable once written.
        iii. Only authenticated users can submit reimbursements.
        iv. Logs must be stored securely for 5 years.
    f. **Conceptual flow:**
        i. The system receives the user request from an API call.
        ii. The system finds the expense and action type and finds related policy and rules.
        iii. The system compares the rules to the expense details and determines if the request is accepted or denied.
        iv. The system comes to a decision and generates a natural language explanation.
        v. The system stores the user ID, action type, expense detail, and date/time to the audit log.
        vi. The system returns the decision and explanation to the user.
2. **As an Employee**, I want to submit reimbursement requests, so I can get paid back quickly and transparently.
    a. **Outcome:** The Employee can submit a reimbursement request and can view whether the submission was successful or why it failed.
    b. **Requirements:**
        i. The system must be able to validate the user.
        ii. The system must be able to take reimbursement request details from the user.
        iii. The system must be able to tell the user if the request succeeded or why it failed.
    c. **Inputs:**
        i. User credentials or a session token (for authentication/authorization).
        ii. Reimbursement request details: action type, expense category, amount, description.
    d. **Outputs:**
        i. A message that the submission was successful or why it failed.
    e. **Constraints:**
        i. The system must validate users submit a reimbursement.
        ii. The system must require all reimbursement request details before submission.
        iii. (Future): The system should prevent duplicate reimbursement submissions for the same item/expense.
    f. **Conceptual flow:**
        i. The system receives the user's credentials (through a session token or by logging in another way).

ii. The system receives the filled out form with all the reimbursement request details.
iii. The system either tells the user it was submitted correctly or that it failed and why.

3. **As a Policy Maker**, I want to view the current list of policies and rules, so I can understand the standards before making changes.
    a. **Outcome:** The Policy Maker can view all policies, filter them by criteria, and drill down into a specific policy to see its rules.
    b. **Requirements:**
        i. The system must validate that the user is a Policy Maker.
        ii. The system must show the user all policies and rules.
        iii. The system must show groupings of policies and rules based on search queries.
        iv. The system must be able to show a specific rule or policy.
    c. Inputs:
        i. User credentials or session token.
        ii. An individual ID of the rule or policy.
        iii. A search term that can be used to retrieve a group of rules or policies.
        iv. A request to return all policies and rules.
    d. Outputs:
        i. An individual policy or rule, a group of them, or all of them.
    e. Constraints:
        i. The system must accept only valid search terms.
        ii. The system must only allow searches by individual rule and policy IDs.
        iii. The system must validate that the user is a Policy Maker.
        iv. The search function will only return rules and policies.
    f. Conceptual Flow:
        i. The system receives the user's credentials (through a session token or by logging in another way).
        ii. The system receives one of the following:
            1. A specific rule or policy by ID.
            2. A group of policies and rules by search term.
            3. All policies and rules.
        iii. The system retrieves the requested information and presents it to the user.
4. **As a Policy Maker**, I want to add new policies and rules, so business regulations can be enforced.
    a. **Outcome:** The Policy Maker can create policies without rules, and create rules to attach to policies.
    b. Requirements:
        i. The system must validate that the user is a Policy Maker.
        ii. The system must create a policy when the user provides valid policy details.

iii. The system must create a rule when the user provides valid rule details and the policy ID to which the new rule will be attached.

c. Inputs:

i. User credentials or session token.
ii. The policy details.
iii. The rule details and the policy ID.

d. Outputs:

i. A message saying the creation was successful or why it failed.

e. Constraints:

i. A rule cannot be made without a policy ID.
ii. The system must validate that the user is a Policy Maker.
iii. The policy ID must be unique.
iv. The rule ID must be unique.

f. Conceptual Flow:

i. The system receives the user's credentials (through a session token or by logging in another way).
ii. *To create a policy:* The system receives all the policy information.
iii. *To create a rule:* The system receives all the rule information and the policy ID.
iv. The system validates the input, creates the policy or rule, or documents why it failed, and informs the user of the result.

5. **As a Policy Maker**, I want to edit existing policies and rules, so I can keep rules current when standards change.

a. Outcome: The Policy Maker can modify existing policies and rules.

b. Requirements:

i. The system must validate that the user is a Policy Maker.
ii. The system must require the ID of the policy or rule to be changed.
iii. The system must update the specific policy or rule based on the valid changes provided by the user.
iv. (Future) The system must retain a record of the prior version of the policy/rule for auditing purposes.

c. Inputs:

i. User credentials or session token.
ii. The policy or rule ID.
iii. The proposed changes to the policy or rule.

d. Outputs:

i. A message saying the edit was successful or why it failed.
ii. (Future) The system also records the updated version.

e. Constraints:

i. The new changes must be valid.
ii. The system must validate that the user is a Policy Maker.
iii. The policy or rule must already exist.
iv. Changes must not remove required fields.

f. Conceptual Flow:

i. The system receives the user's credentials (through a session token or by logging in another way).
ii. The system receives the ID of the policy or rule to be change
iii. The system receives the changes they want to make.
iv. The system validates the changes, updates the specific policy or rule, and informs the user of the result.

6. **As a Policy Maker**, I want to delete outdated policies, so they don't interfere with current reimbursement decisions.
    a. **Outcome:** The Policy Maker can delete individual rules, and can delete policies that have no rules attached.
    b. **Requirements:**
        i. The system must validate that the user is a Policy Maker.
        ii. The system must delete a rule when the user provides a valid rule ID.
        iii. The system must delete the chosen policy that has no rules when the user provides a valid policy ID.
    c. **Inputs:**
        i. User credentials or session token.
        ii. The policy or rule ID.
    d. **Outputs:**
        i. A message saying the deletion was successful or why it failed.
    e. **Constraints:**
        i. The system must verify that the policy or rule exists.
        ii. The system must verify that the policy has no attached rules.
        iii. The system must validate that the user is a Policy Maker.
    f. **Conceptual Flow:**
        i. The system receives the user's credentials (through a session token or by logging in another way).
        ii. The system receives the ID of the policy or rule to be deleted.
        iii. The system validates the request, attempts to delete the specified policy or rule, and informs the user whether the deletion was successful or why it failed.
7. **As an Auditor,** I want to retrieve and filter logs of all decisions and actions, so I can investigate specific cases and verify compliance efficiently.
    a. **Outcome:** The Auditor can view audit logs for specific users or filter them by criteria such as date range, decision type, or policy ID. The Auditor can optionally export the filtered logs.
    b. **Requirements:**
        i. The system must validate that the user is an Auditor.
        ii. The system must retrieve all audit log entries associated with the provided user ID, a decision type, a date range, or a policy ID.
        iii. The system must present the logs in a readable format.
        iv. (Future) The system must allow the Auditor to export retrieved logs in a chosen format (e.g., CSV, PDF).
    c. **Inputs:**

i. User credentials or session token.
ii. One of the following filters:
    1. A user ID.
    2. A decision type.
    3. A date range.
    4. A policy ID.

d. Outputs:

i. The system must return a list of logs based on the user's filter (a user ID, a decision type, a date range, or a policy ID).
ii. (Future) The system must be able to export the logs in a readable format.

e. Constraints:

i. The system must validate that the user is an Auditor.
ii. The system must require at least one valid filter:
    1. A valid user ID.
    2. A valid date range.
    3. A valid decision type.
    4. A valid policy ID.
iii. The requested logs must exist.
iv. (Future) Auditors may only view logs for users they are authorized to audit.

f. Conceptual Flow:

i. The system receives the user's credentials (through a session token or by logging in another way).
ii. The system receives one or more filters (user ID, date range, decision type, policy ID).
iii. The system retrieves and displays all logs that match the filters.
iv. (Future) The system receives the request to export the data.
v. The system exports the logs in a readable format.

System Requirements:

Security:

1. **Authentication**: The system must validate users before granting access.
2. **Authorization**: The system must authorize users based on their role before performing actions.
3. **Data Protection**: The system must encrypt sensitive data in storage and in transit.
4. **Session Management**: The system must automatically log users out after their session expires.
5. **Audit & Access Logging**: The system must log all user actions and access attempts, and store these logs securely for auditing.

Automation:

1. **Policy Evaluation**: The system must evaluate each submission against all applicable rules under the relevant policy.
2. **Decision Generation**: The system must produce an accept or deny decision by comparing the submission to the relevant policy.
3. **Explanation Generation**: The system must generate a natural language explanation for its decision.
4. **Audit Logging**: The system must ensure all logged data is immutable and securely stored.
5. **Error Handling**: The system must log all errors and provide meaningful error messages without exposing sensitive details.

Reliability / Performance:

1. **Availability**: The system must be available to authorized users with minimal downtime.
2. **Response Time**: The system must return a decision to user requests within a reasonable time frame.
3. **Scalability**: The system must be able to handle increased numbers of users and submissions without performance degradation.
4. **Fault Tolerance**: The system must continue to operate or recover gracefully in the event of system or component failures.
5. **Data Integrity**: The system must ensure that no data is lost, duplicated, or corrupted during processing or storage.

Data Management:

1. **Data Retention**: The system must retain audit logs and policy records for a defined period to support compliance and review.
2. **Backups**: The system must perform regular backups of all critical data and store them securely.
3. **Version Control**: The system must maintain version history of policies and rules to support auditing and rollback.
4. **Consistency**: The system must ensure that data is consistent across all components, even in concurrent access scenarios.
5. **Data Deletion**: The system must securely delete data when it is no longer required or when retention periods expire.

Future Considerations:

1. Manager approval workflows for exceptional cases.

2. Duplicate submission detection to prevent multiple reimbursements for the same expense.

3. Advanced reporting and analytics tools.

4. Integration with third-party financial or HR systems.