# **Appetite Checker Solution for Insurance**

This document outlines the design and implementation structure for an Appetite Checker solution, a SaaS product designed to integrate with agent/broker portals. This solution aims to validate submission data against insurance carrier appetite rules, predicting approval or rejection and preventing the submission of un-approvable applications. This will save time and resources by filtering out submissions that are unlikely to be approved.

# **1. System Architecture Diagram**

# 

# **2. Component Description**

* **Agent/Broker Portal:** Existing portal used by agents and brokers to submit insurance applications. This will integrate with the Appetite Checker Web UI.
* **Appetite Checker Web UI:** A web-based interface embedded within the Agent/Broker Portal. It allows agents to input or upload application data and trigger the appetite check. It displays the results (approval/rejection prediction) and provides reasons for the decision.
* **API Gateway:** A central point of entry for all requests to the Appetite Checker Service. It handles routing, rate limiting, and security.
* **Authentication/Authorization Service:** Authenticates users and authorizes access to the Appetite Checker Service. This ensures that only authorized agents and brokers can use the system.
* **Appetite Checker Service:** The core service that receives application data, invokes the Rule Engine, and returns the approval/rejection prediction. It also interacts with external data sources.
* **Rule Engine:** Evaluates the application data against the rules defined in the Carrier Appetite Database. It determines whether the application meets the carrier's appetite. A rules engine like Drools or a custom-built engine can be used.
* **Carrier Appetite Database:** Stores the rules and criteria that define each carrier's appetite. This database needs to be regularly updated to reflect changes in carrier preferences.
* **Reporting/Analytics Service:** Collects data on appetite check requests, results, and user activity. This data is used to generate reports and dashboards that provide insights into the effectiveness of the Appetite Checker.
* **Dashboard:** Provides a visual representation of key metrics, such as the number of appetite checks performed, the approval/rejection rate, and the reasons for rejection.
* **External Data Sources:** Third-party data sources that provide additional information about the applicant or the risk being insured. Examples include credit scores, property data, and industry-specific risk data.
* **Admin Portal:** A web-based interface for administrators to manage the system, including updating rules, managing users, and monitoring performance.
* **Rule Management UI:** A component within the Admin Portal that allows administrators to create, update, and delete rules in the Carrier Appetite Database.

# **3. Data Flow**

1. **Agent/Broker initiates Appetite Check:** The agent/broker enters application data into the Agent/Broker Portal and clicks a button to initiate the appetite check.
2. **Data Submission to Web UI:** The application data is sent to the Appetite Checker Web UI.
3. **Request to API Gateway:** The Web UI sends a request to the API Gateway, including the application data and the agent's authentication token.
4. **Authentication and Authorization:** The API Gateway verifies the agent's authentication token with the Authentication/Authorization Service.
5. **Request Routing to Appetite Checker Service:** The API Gateway routes the request to the Appetite Checker Service.
6. **Data Enrichment (Optional):** The Appetite Checker Service may enrich the application data by querying External Data Sources.
7. **Rule Engine Invocation:** The Appetite Checker Service invokes the Rule Engine, passing the application data.
8. **Rule Evaluation:** The Rule Engine evaluates the application data against the rules in the Carrier Appetite Database.
9. **Decision and Explanation:** The Rule Engine returns a decision (approval/rejection) and an explanation of the reasons for the decision.
10. **Response to Web UI:** The Appetite Checker Service returns the decision and explanation to the Appetite Checker Web UI.
11. **Display Results:** The Web UI displays the results to the agent/broker.
12. **Data Logging:** The Reporting/Analytics Service logs the request, the application data, the decision, and the explanation.

# **4. Implementation Structure**

* **Technology Stack:**
  + **Frontend (Web UI):** React, Angular, or Vue.js
  + **Backend (API Gateway, Appetite Checker Service, Authentication/Authorization Service, Reporting/Analytics Service):** Java (Spring Boot), Python (Flask/Django), or Node.js (Express)
  + **Rule Engine:** Drools, jRule, or a custom-built engine
  + **Database (Carrier Appetite Database):** PostgreSQL, MySQL, or MongoDB
  + **Cloud Platform:** AWS, Azure, or Google Cloud Platform
* **Development Process:**
  + Agile development methodology with short sprints and frequent releases.
  + Automated testing (unit, integration, and end-to-end) to ensure quality.
  + Continuous integration and continuous deployment (CI/CD) to automate the build, test, and deployment process.
* **Security Considerations:**
  + Secure authentication and authorization to protect sensitive data.
  + Data encryption at rest and in transit.
  + Regular security audits and penetration testing.
  + Compliance with relevant data privacy regulations (e.g., GDPR, CCPA).
* **Scalability and Performance:**
  + Use of a microservices architecture to allow for independent scaling of individual components.
  + Caching to reduce database load and improve response times.
  + Load balancing to distribute traffic across multiple servers.
  + Monitoring and alerting to identify and address performance bottlenecks.

# **5. Integration with Agent/Broker Portal**

The Appetite Checker Web UI will be integrated into the existing Agent/Broker Portal as an iframe or a separate module. The integration will involve:

* **Authentication:** Sharing authentication tokens between the Agent/Broker Portal and the Appetite Checker Service. This can be achieved using standards like OAuth 2.0 or SAML.
* **Data Exchange:** Defining a clear data format for exchanging application data between the Agent/Broker Portal and the Appetite Checker Service (e.g., JSON).
* **User Experience:** Ensuring a seamless user experience by matching the look and feel of the Appetite Checker Web UI to the Agent/Broker Portal.

# **6. Rule Management**

The Rule Management UI will allow administrators to:

* **Create new rules:** Define the criteria for approval or rejection based on various application data fields.
* **Update existing rules:** Modify the criteria for existing rules.
* **Delete rules:** Remove rules that are no longer relevant.
* **Test rules:** Evaluate rules against sample application data to ensure they are working as expected.
* **Version control:** Track changes to rules over time.
* **Rule deployment:** Deploy new or updated rules to the Carrier Appetite Database.

# **7. Reporting and Analytics**

The Reporting/Analytics Service will collect data on:

* **Appetite check requests:** The number of requests received.
* **Approval/rejection rates:** The percentage of applications that are approved or rejected.
* \*\*Reasons for rejection