## Business Requirements Document: Premium Calculation Engine (for Lucienne)

1. Introduction

* 1.1 Purpose: This document outlines the business requirements for the Premium Calculation Engine application. It describes the application's purpose, target audience, key features, and overall business goals.
* 1.2 Scope: This BRD covers all aspects of the application, including user authentication, data management, premium calculation, user management, and deployment.
* 1.3 Target Audience: This document is intended for stakeholders, developers, project managers, and testers involved in the development and deployment of the Premium Calculation Engine.

2. Business Context

* 2.1 Background: The Premium Calculation Engine addresses the need for a centralized, efficient, and accurate system for calculating insurance premiums. Currently, the process is manual and prone to errors, leading to inefficiencies and potential financial losses.
* 2.2 Business Goals:
  + Improve the accuracy of premium calculations.
  + Increase efficiency in the premium calculation process.
  + Provide a user-friendly interface for managing product rates and related data.
  + Enable secure access and user management.
* 2.3 Stakeholders:
  + Insurance Underwriters
  + Actuarial Team
  + IT Department
  + Administrators

3. Functional Requirements

* 3.1 User Authentication:
  + Users can sign up for an account using their email address and a secure password.
  + Users can log in to the application using their credentials.
  + Users can reset their password if they forget it.
  + The system supports different user roles (e.g., user, admin) with varying levels of access.
* 3.2 Data Management:
  + Administrators can manage the following data tables:
    - Product Rate Master: Base product rates, health categories, and rebate settings.
    - Product Rate Detail: Detailed product rates by scale and payment frequency.
    - Rebate Percentage: Rebate percentages by income tier and rebate type.
    - Scale Factors: Scale factors for different product scales.
    - Risk Loading: Risk loading factors based on age and sex.
  + Administrators can add, edit, and delete records in each table.
  + The application supports uploading data from spreadsheet files (.xlsx).
  + The application validates data inputs to ensure accuracy and consistency.
* 3.3 Premium Calculation:
  + The application provides a "Premium Test" feature that allows users to calculate premiums based on various parameters.
  + Users can input the following parameters:
    - Effective Date
    - Product Code(s)
    - State Code
    - Scale Code
    - Rate Code
    - Payment Frequency
    - Rebate Type
    - LHC Percentage
    - Risk Rating (Age and Sex)
  + The application calculates the premium based on the data in the data tables and the user-provided parameters.
  + The application displays a detailed breakdown of the premium calculation, including:
    - Base Premium
    - Scale Factor
    - Scale and Frequency Premium
    - Risk Loading Amount(s)
    - LHC Amount
    - Rebate Amount
    - Final Premium
* 3.4 User Management:
  + Administrators can view a list of all users.
  + Administrators can assign roles to users (e.g., user, admin).
  + Administrators can reset user passwords.
  + Administrators can delete user accounts.

4. Non-Functional Requirements

* 4.1 Performance:
  + The application should load quickly and respond to user interactions in a timely manner.
  + Premium calculations should be performed efficiently.
* 4.2 Security:
  + The application must protect sensitive data, such as user credentials and financial information.
  + Access to data and functionality should be restricted based on user roles.
  + The application should be protected against common web vulnerabilities (e.g., XSS, SQL injection).
* 4.3 Usability:
  + The application should have a user-friendly interface that is easy to navigate and understand.
  + The application should provide clear and concise error messages.
* 4.4 Reliability:
  + The application should be reliable and available when needed.
  + The application should handle errors gracefully and prevent data loss.
* 4.5 Scalability:
  + The application should be scalable to accommodate a growing number of users and data.

5. Data Requirements

* 5.1 Data Tables:
  + ProductRateMaster
  + Columns:
  + ProductCode (text, not null)
  + StateCode (text, not null)
  + RateCode (integer, not null)
  + DateOn (date, not null)
  + DateOff (date, nullable)
  + HealthCategory (smallint, nullable)
  + BaseRate (double precision, nullable)
  + LHCApplicable (text, nullable)
  + RebateApplicable (text, nullable)
  + LastUpdateTimestamp (timestamp with time zone, default: now())
  + LastUpdateUser (character varying, nullable)
  + Indexes:
  + ProductRateMaster\_pkey (unique, btree)
  + Constraints:
  + PRIMARY KEY (ProductCode, StateCode, RateCode, DateOn)
  + Policies:
  + CRUD policies for authenticated users
  + RiskLoading
  + Columns:
  + ProductCode (text, not null)
  + DateOn (date, not null)
  + DateOff (date, nullable)
  + Sex (text, not null)
  + Age (integer, not null)
  + RiskLoading (real, nullable)
  + LastUpdateUser (text, nullable)
  + LastUpdateTimestamp (timestamp without time zone, nullable)
  + Indexes:
  + RiskLoading\_pkey (unique, btree)
  + Constraints:
  + PRIMARY KEY (ProductCode, DateOn, Sex, Age)
  + Policies:
  + CRUD policies for authenticated users
  + ScaleFactors
  + Columns:
  + ProductCode (text, not null)
  + ScaleCode (text, not null)
  + DateOn (date, not null)
  + DateOff (date, nullable)
  + ScaleFactor (real, nullable)
  + LastUpdateUser (text, nullable)
  + LastUpdateTimestamp (text, nullable)
  + Indexes:
  + ScaleFactors\_pkey (unique, btree)
  + Constraints:
  + PRIMARY KEY (ProductCode, ScaleCode, DateOn)
  + Policies:
  + CRUD policies for authenticated users
  + RebatePercentage
  + Columns:
  + RebateType (text, not null)
  + IncomeTier (smallint, not null)
  + DateOn (date, not null)
  + DateOff (date, nullable)
  + Rebate (real, nullable)
  + LastUpdateUserid (text, default: COALESCE((jwt() ->> 'email'::text), 'system'::text))
  + LastUpdateTimestamp (timestamp without time zone, nullable)
  + Indexes:
  + RebatePercentage\_pkey (unique, btree)
  + Constraints:
  + PRIMARY KEY (RebateType, IncomeTier, DateOn)
  + Policies:
  + CRUD policies for authenticated users
  + profiles
  + Columns:
  + id (uuid, primary key, references auth.users)
  + email (text, unique, not null)
  + role (text, not null, default: 'user')
  + created\_at (timestamp with time zone, default: now())
  + last\_sign\_in\_at (timestamp with time zone, nullable)
  + Indexes:
  + idx\_profiles\_email (btree)
  + idx\_profiles\_role (btree)
  + profiles\_email\_key (unique, btree)
  + profiles\_pkey (unique, btree)
  + Constraints:
  + UNIQUE (email)
  + PRIMARY KEY (id)
  + CHECK (role = ANY (ARRAY['user'::text, 'admin'::text]))
  + Foreign Keys:
  + FOREIGN KEY (id) REFERENCES users(id) ON DELETE CASCADE
  + Policies:
  + Policies for user and admin access control
  + ProductRateDetail
  + Columns:
  + ProductCode (text, not null)
  + StateCode (text, not null)
  + RateCode (integer, not null)
  + ScaleCode (text, not null)
  + DateOn (date, not null)
  + DateOff (date, nullable)
  + WeeklyRate (double precision, nullable)
  + MonthlyRate (double precision, nullable)
  + QuarterlyRate (double precision, nullable)
  + HalfYearlyRate (double precision, nullable)
  + YearlyRate (double precision, nullable)
  + LastUpdateUser (text, nullable)
  + LastUpdateTimestamp (timestamp without time zone, default: now())
  + Indexes:
  + ProductRateDetail\_pkey (unique, btree)
  + Constraints:
  + PRIMARY KEY (ProductCode, StateCode, RateCode, ScaleCode, DateOn)
  + Policies:
  + CRUD policies for authenticated users
* 5.2 Data Validation:
  + Product Code: 3 characters
  + State Code: Valid state code from the list of available codes
  + Rate Code: Integer between 0 and 999
  + Dates: Valid date format (YYYY-MM-DD)
  + Health Category: 2, 3, or 5
  + Base Rate: Non-negative number
  + LHC Applicable: 'Y' or 'N'
  + Rebate Applicable: 'Y' or 'N'
  + Scale Code: Valid scale code from the list of available codes
  + Risk Loading: Non-negative number between 0 and 10
  + Rebate: Non-negative number between 0 and 45
  + Age: Integer between 0 and 120

6. Deployment Requirements

* 6.1 Environment:
  + The application should be deployed to a cloud-based platform (e.g., Netlify).
* 6.2 Scalability:
  + The deployment environment should be scalable to handle increasing traffic and data volume.
* 6.3 Maintenance:
  + The deployment process should be automated to facilitate updates and maintenance.

7. Future Enhancements

* Integration with other insurance systems.
* Advanced reporting and analytics.
* Customizable premium calculation rules.
* API for external access.

This BRD provides a comprehensive overview of the requirements for the Premium Calculation Engine application. It should be used as a guide for development, testing, and deployment.