Credit Score Calculation Logic - Technical Specification

This document outlines the comprehensive credit scoring system used to evaluate small business loan applications. The system calculates scores across five key categories and combines them into an overall credit rating.

Summary

The credit scoring system evaluates applicants across five categories with the following weights:

- **Financial Health (35%)**: Evaluates debt service capability and financial stability
- **Credit History (25%)**: Assesses past credit behavior and repayment history
- **Business Stability (20%)**: Measures business longevity and operational scale
- **Operational Efficiency (10%)**: Analyzes business processes and digital presence
- **Risk & Support Factors (10%)**: Considers industry risk and collateral

Each category is scored on a 0-100 scale, then combined using weighted averaging to produce an overall score. The final rating is determined as follows:

- **Good**: 85-100
- **Average**: 70-84
- **Bad**: 55-69
- **Poor**: 0-54

Detailed Calculation Methodology

Below is a deep-dive into the two steps that happen inside `CreditReport.tsx` when you press "View Report":

1. `calculateCategoryScores(allInputs)` – gives one score (0-100) for every form-page / category.
2. `calculateOverallScore(subScores)` – folds those five numbers into a single overall score (0-100) and converts it into a rating badge (Good / Average / Bad / Poor).

Everything lives in `src/services/creditScoreCalculator.ts`, which is a set of pure functions – no React, no stores, just maths.

---

1. Category (page-level) scores

Each function starts with a **baseline of 50 points**, adds bonuses and subtracts penalties, then calls the helper `clamp()` so that the result is always between 0 and 100.

A. Financial – filled on the “Store Details” page

```
// Financial Score Calculation
const financialScore = (f: FinancialInputs): number => {
  const debtRatio = f.monthlySales === 0 ? 100 : (f.monthlyEMI /
f.monthlySales) * 100;
  let score = 50;
  score += debtRatio <= 30 ? 20 : debtRatio <= 50 ? 10 : 0;
  score += Math.min(f.profitMargin * 2, 20);
  score += Math.min(f.averageBankBalance / 10_000, 10);
  score += f.buildingOwnership === 'own' ? 10 : 0;
  score += f.itrFiled ? 10 : 0;
  return clamp(score);
};
```

```
•  Maximum bonus path ⇒ 50 + 20 + 20 + 10 + 10 + 10 = 120 → clamped to
100.
•  Biggest impact driver is the EMI-to-sales **debt ratio** (up to 20 pts).

B. Credit History – “Credit History” page
```typescript
// Credit History Score Calculation
const creditHistoryScore = (c: CreditHistoryInputs): number => {
  let score = c.cibilScore ? (c.cibilScore - 300) / 5.5 : 50;
  score -= Math.min(c.pastLoanDefaults * 10, 50);
  score -= Math.min(c.returnedCheques * 5, 20);
  score -= Math.min(c.loanApplications * 5, 25);
  score += Math.min(c.bankingRelationship * 2, 20);
  score += Math.min(c.fullyRepaidLoans * 5, 15);
  return clamp(score);
};
```

```
•  A high CIBIL starts you high; defaults can wipe up to 50 pts.

C. Business Stability – “Business Stability” page
```typescript
// Business Stability Score Calculation
const businessStabilityScore = (b: BusinessStabilityInputs): number => {
  let score = 50;
  score += Math.min(b.yearsInOperation * 2, 20);
  score += Math.min(b.annualRevenue / 1_000_000, 20);
  score += Math.min(b.numberOfEmployees / 5, 10);
  score += Math.min(b.shopSize / 100, 10);
  score += Math.min(b.numberOfBranches * 2, 10);
  score += b.sellsPrivateLabel ? 5 : 0;
  return clamp(score);
};
```

D. Operational Efficiency – “Operational” page
```typescript
// Operational Efficiency Score Calculation
const operationalScore = (o: OperationalInputs): number => {
  const digitalPayments = o.digitalPaymentsAdoption ?? 0;
  const inventory = o.inventoryTurnover ?? 'monthly';
  const seasonal = o.seasonalImpact ?? 'none';
  const footfall = o.averageMonthlyFootfall ?? 0;
  const timings = o.shopTimings ?? 0;
  const online = o.onlinePresence ?? { socialMedia: false, website: false,
ecommerce: false };

  let score = 50;
  score += Math.min(digitalPayments, 20);
  score += inventory === 'weekly' ? 20 : inventory === 'monthly' ? 10 :
inventory === 'quarterly' ? -10 : -20;
  score += seasonal === 'none' ? 10 : seasonal === 'low' ? 5 : seasonal ===
'medium' ? -5 : -10;
  score += footfall >= 3000 ? 10 : footfall >= 1000 ? 5 : 0;

  const onlineBonus = (online.socialMedia ? 5 : 0) + (online.website ? 5 :
0) + (online.ecommerce ? 10 : 0);
  score += Math.min(onlineBonus, 15);
  score += timings >= 12 ? 10 : timings >= 10 ? 5 : 0;

  return clamp(score);
};
```

Weekly inventory turns plus high digital payments alone can already add 40 pts.

E. Risk & Support – “Risk & Support Factors” page
```typescript
// Risk & Support Score Calculation
const riskSupportScore = (r: RiskSupportInputs): number => {
  let score = 50;
  score += r.distributorPaymentRegularity ? 10 : -10;
  score += r.industryType === 'grocery' || r.industryType === 'pharmacy' ?
10 :
           r.industryType === 'clothing' || r.industryType === 'restaurant'
? -10 : 0;
  score += r.purposeOfLoan === 'growth' ? 5 : r.purposeOfLoan ===
'refinance' ? -5 : 0;

  if (r.collateralProvided && r.collateralValue && r.loanAmountRequested >
0) {
    const ratio = r.collateralValue / r.loanAmountRequested;
    score += ratio >= 2 ? 15 : ratio >= 1.5 ? 10 : ratio >= 1 ? 5 : 0;
  } else if (!r.collateralProvided) {
    score -= 10;
```

```
  }

  return clamp(score);
};
```

```
---------------------------------------------------------------
2.  Overall score & rating
---------------------------------------------------------------
```typescript
// Overall Score Calculation
export const calculateOverallScore = (scores: CategoryScores) => {
  const total =
    scores.financial * 0.35 +
    scores.creditHistory * 0.25 +
    scores.businessStability * 0.2 +
    scores.operational * 0.1 +
    scores.riskSupport * 0.1;

  let rating: 'Good' | 'Average' | 'Bad' | 'Poor';
  if (total >= 85) rating = 'Good';
  else if (total >= 70) rating = 'Average';
  else if (total >= 55) rating = 'Bad';
  else rating = 'Poor';

  return { total: Math.round(total), rating } as const;
};
```

---

3. End-to-end data flow

End-to-end Data Flow

1. **Data Collection**: Each form page collects and stores user inputs in a Zustand store.
2. **Input Aggregation**: The `CreditReport` component aggregates all inputs into a single `allInputs` object.
3. **Category Scoring**: `calculateCategoryScores(allInputs)` processes the inputs to generate scores for each category, returning an object like:

```
{
  "financial": 78,
  "creditHistory": 66,
  "businessStability": 72,
  "operational": 85,
  "riskSupport": 60
}
```
```

4. **Score Visualization**: Category scores are displayed in the "Score Breakdown" section using mini-cards.
5. **Overall Calculation**: The category scores are fed into `calculateOverallScore()`, which:
    - Applies the category weights
    - Calculates a weighted average
    - Rounds to the nearest integer
    - Determines the appropriate rating (Good/Average/Bad/Poor)
6. **Result Display**: The final score and rating are displayed in the large circular meter and rating chip.

Key Implementation Notes

- **Baseline Scoring**: Each category starts with a baseline score of 50 points.
- **Rule-based Adjustments**: Points are added or subtracted based on specific business rules for each category.
- **Score Clamping**: All category scores are clamped between 0-100 to maintain consistency.
- **Weighted Composition**: The overall score is a weighted average of all category scores.
- **Type Safety**: The implementation uses TypeScript for type safety and better developer experience.

This scoring system provides a comprehensive evaluation of a business's creditworthiness by considering multiple financial and operational factors.