Contents

SECTION 43: BILLING & CREDITS SYSTEM (v4.13.0)

Version: 4.13.0 | Adds 7-tier subscriptions and prepaid credits system

---

43.1 BILLING SYSTEM OVERVIEW

```
RADIANT v4.13.0 - BILLING SYSTEM


  7-TIER SUBSCRIPTION MODEL
  (Similar to OpenAI/ChatGPT)

• FREE - Trial ($0, 0.5 credits)
• INDIVIDUAL - Personal ($29/mo)
• FAMILY - Household ($49/mo, 5 users)
• TEAM - Small biz ($25/user, 2-25)
• BUSINESS - Mid-market ($45/user)
• ENTERPRISE - Large orgs (Custom)
• ENTERPRISE PLUS - Compliance incl.


  PREPAID CREDITS SYSTEM
  (Similar to Windsurf Pro)

• $10 = 1 Credit (any quantity)
```

- Volume discounts (5% to 25% off)
- Bonus credits for bulk purchases
- Credit pools for families/teams
- Auto-purchase when balance low
- Credits never expire

COMPLIANCE ADD-ON ($25/user/mo)
Available for TEAM+ tiers

- HIPAA + BAA
- SOC 2 Type II
- GDPR/CCPA compliance
- Customer-managed encryption keys
- Enhanced audit logging
- Data residency options

43.2 SUBSCRIPTION TIERS

packages/shared/src/billing/tiers.ts

```typescript
/**
 * RADIANT Subscription Tiers
 * 7-tier model from FREE to ENTERPRISE PLUS
 *
 * @version 4.13.0
 * @module @radiant/shared/billing
 */

// =============================================================================
// TIER DEFINITIONS
// =============================================================================

export type SubscriptionTierId =
  | 'FREE'
  | 'INDIVIDUAL'
  | 'FAMILY'
  | 'TEAM'
  | 'BUSINESS'
  | 'ENTERPRISE'
  | 'ENTERPRISE_PLUS';

export interface SubscriptionTier {
```

```
  id: SubscriptionTierId;
  displayName: string;  // Fallback - use localizationKey at runtime
  description: string;  // Fallback - use localizationKey at runtime
  localizationKey: string;  // e.g., 'billing.tiers.free' for i18n lookup

  // Pricing
  pricing: {
    monthly: number | null;           // null = contact sales
    annual: number | null;
    perUser: boolean;
    minUsers?: number;
    maxUsers?: number;
    currency: string;
    contactSales?: boolean;
  };

  // Credits
  includedCreditsPerUser: number;     // Monthly credits per user/seat

  // Rate limits
  rateLimits: {
    requestsPerMinute: number;
    tokensPerMinute: number;
    concurrentRequests: number;
  };

  // Features
  features: TierFeatures;

  // Add-ons available
  availableAddOns: string[];

  // Display
  isPublic: boolean;
  sortOrder: number;
  badge?: string;
}

export interface TierFeatures {
  modelAccess: 'limited' | 'full' | 'full_plus_beta' | 'all';
  maxModelsPerMonth: number | null;
  maxRequestsPerDay: number | null;
  maxTokensPerRequest: number;
  priorityQueue: boolean;
  apiAccess: boolean;
  exportFormats: string[];
  supportLevel: 'community' | 'email' | 'priority_email' | 'priority' | 'dedicated';
  dataRetention: string;
```

```
  watermarkedOutputs: boolean;
  conversationHistory: boolean;
  customInstructions: boolean;

  // Sharing features
  sharedCreditPool: boolean;
  teamWorkspaces: boolean;
  sharedConversations: boolean;

  // Admin features
  teamAdminDashboard: boolean;
  usageAnalytics: boolean;
  roleBasedAccess: boolean;
  inviteManagement: boolean;

  // Enterprise features
  ssoIntegration: boolean;
  scimProvisioning: boolean;
  auditLogs: boolean;
  dataExport: boolean;
  customBranding: boolean;
  dedicatedAccountManager: boolean;
  apiRateLimitCustomization: boolean;
  webhooks: boolean;
  ipWhitelisting: boolean;

  // Family features
  parentalControls?: boolean;
  familyAdminDashboard?: boolean;
  perMemberUsageLimits?: boolean;

  // Enterprise Plus features
  slaGuarantee?: boolean;
  uptimeGuarantee?: string;
  customModelFineTuning?: boolean;
  dedicatedInfrastructure?: boolean;
  multiRegionDeployment?: boolean;
  onPremiseDeployment?: boolean;
}

// ============================================================================
// TIER REGISTRY
// ============================================================================

export const SUBSCRIPTION_TIERS: Record<SubscriptionTierId, SubscriptionTier> = {
  FREE: {
    id: 'FREE',
    displayName: 'Free Trial',  // Fallback
```

```
  description: 'Try RADIANT with limited features',  // Fallback
  localizationKey: 'billing.tiers.free',
  pricing: { monthly: 0, annual: 0, perUser: false, currency: 'USD' },
  includedCreditsPerUser: 0.5,
  rateLimits: { requestsPerMinute: 10, tokensPerMinute: 20_000, concurrentRequests: 1 },
  features: {
    modelAccess: 'limited', maxModelsPerMonth: 5, maxRequestsPerDay: 50,
    maxTokensPerRequest: 4_000, priorityQueue: false, apiAccess: false,
    exportFormats: ['txt'], supportLevel: 'community', dataRetention: '7_days',
    watermarkedOutputs: true, conversationHistory: false, customInstructions: false,
    sharedCreditPool: false, teamWorkspaces: false, sharedConversations: false,
    teamAdminDashboard: false, usageAnalytics: false, roleBasedAccess: false,
    inviteManagement: false, ssoIntegration: false, scimProvisioning: false,
    auditLogs: false, dataExport: false, customBranding: false,
    dedicatedAccountManager: false, apiRateLimitCustomization: false,
    webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: [],
  isPublic: true,
  sortOrder: 0,
},

INDIVIDUAL: {
  id: 'INDIVIDUAL',
  displayName: 'Individual',  // Fallback
  description: 'Full-featured personal plan',  // Fallback
  localizationKey: 'billing.tiers.individual',
  pricing: { monthly: 29, annual: 290, perUser: false, currency: 'USD' },
  includedCreditsPerUser: 3,
  rateLimits: { requestsPerMinute: 60, tokensPerMinute: 100_000, concurrentRequests: 5 },
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: 1_000,
    maxTokensPerRequest: 128_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx'], supportLevel: 'email',
    dataRetention: '90_days', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: false, teamWorkspaces: false,
    sharedConversations: false, teamAdminDashboard: false, usageAnalytics: false,
    roleBasedAccess: false, inviteManagement: false, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: ['API_POWER_PACK'],
  isPublic: true,
  sortOrder: 1,
},

FAMILY: {
```

```
  id: 'FAMILY',
  displayName: 'Family',  // Fallback
  description: 'Share AI across your household',  // Fallback
  localizationKey: 'billing.tiers.family',
  id: 'FAMILY',
  displayName: 'Family',
  description: 'Share AI across your household',
  pricing: { monthly: 49, annual: 490, perUser: false, maxUsers: 5, currency: 'USD' },
  includedCreditsPerUser: 6,
  rateLimits: { requestsPerMinute: 120, tokensPerMinute: 200_000, concurrentRequests: 10 },
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: 2_000,
    maxTokensPerRequest: 128_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx'], supportLevel: 'email',
    dataRetention: '90_days', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: false,
    sharedConversations: false, teamAdminDashboard: false, usageAnalytics: false,
    roleBasedAccess: false, inviteManagement: true, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
    parentalControls: true, familyAdminDashboard: true, perMemberUsageLimits: true,
  },
  availableAddOns: ['API_POWER_PACK'],
  isPublic: true,
  sortOrder: 2,
},

TEAM: {
  id: 'TEAM',
  displayName: 'Team',
  description: 'Collaborate with your small team',
  pricing: { monthly: 25, annual: 250, perUser: true, minUsers: 2, maxUsers: 25, currency: '
  includedCreditsPerUser: 3,
  rateLimits: { requestsPerMinute: 300, tokensPerMinute: 500_000, concurrentRequests: 20 },
  features: {
    modelAccess: 'full', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 200_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html'], supportLevel: 'priority_email',
    dataRetention: '1_year', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: true,
    sharedConversations: true, teamAdminDashboard: true, usageAnalytics: true,
    roleBasedAccess: true, inviteManagement: true, ssoIntegration: false,
    scimProvisioning: false, auditLogs: false, dataExport: false,
    customBranding: false, dedicatedAccountManager: false,
    apiRateLimitCustomization: false, webhooks: false, ipWhitelisting: false,
  },
  availableAddOns: ['COMPLIANCE_ADDON', 'PRIORITY_SUPPORT'],
```

```
  isPublic: true,
  sortOrder: 3,
  badge: 'Most Popular',
},

BUSINESS: {
  id: 'BUSINESS',
  displayName: 'Business',
  description: 'Enterprise features for growing companies',
  pricing: { monthly: 45, annual: 450, perUser: true, minUsers: 5, maxUsers: 150, currency:
  includedCreditsPerUser: 5,
  rateLimits: { requestsPerMinute: 1_000, tokensPerMinute: 2_000_000, concurrentRequests: 50
  features: {
    modelAccess: 'full_plus_beta', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 500_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json'], supportLevel: 'priority',
    dataRetention: '2_years', watermarkedOutputs: false, conversationHistory: true,
    customInstructions: true, sharedCreditPool: true, teamWorkspaces: true,
    sharedConversations: true, teamAdminDashboard: true, usageAnalytics: true,
    roleBasedAccess: true, inviteManagement: true, ssoIntegration: true,
    scimProvisioning: true, auditLogs: true, dataExport: true,
    customBranding: true, dedicatedAccountManager: false,
    apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
  },
  availableAddOns: ['COMPLIANCE_ADDON', 'PRIORITY_SUPPORT'],
  isPublic: true,
  sortOrder: 4,
},

ENTERPRISE: {
  id: 'ENTERPRISE',
  displayName: 'Enterprise',
  description: 'Full-scale enterprise deployment',
  pricing: { monthly: null, annual: null, perUser: true, minUsers: 50, currency: 'USD', conta
  includedCreditsPerUser: 10,
  rateLimits: { requestsPerMinute: 5_000, tokensPerMinute: 10_000_000, concurrentRequests: 20
  features: {
    modelAccess: 'all', maxModelsPerMonth: null, maxRequestsPerDay: null,
    maxTokensPerRequest: 1_000_000, priorityQueue: true, apiAccess: true,
    exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json', 'csv'],
    supportLevel: 'dedicated', dataRetention: 'custom', watermarkedOutputs: false,
    conversationHistory: true, customInstructions: true, sharedCreditPool: true,
    teamWorkspaces: true, sharedConversations: true, teamAdminDashboard: true,
    usageAnalytics: true, roleBasedAccess: true, inviteManagement: true,
    ssoIntegration: true, scimProvisioning: true, auditLogs: true, dataExport: true,
    customBranding: true, dedicatedAccountManager: true,
    apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
    slaGuarantee: true, uptimeGuarantee: '99.9%', customModelFineTuning: true,
```

```
      multiRegionDeployment: true,
    },
    availableAddOns: ['COMPLIANCE_ADDON'],
    isPublic: true,
    sortOrder: 5,
  },

  ENTERPRISE_PLUS: {
    id: 'ENTERPRISE_PLUS',
    displayName: 'Enterprise Plus',
    description: 'Maximum security and compliance for regulated industries',
    pricing: { monthly: null, annual: null, perUser: true, minUsers: 100, currency: 'USD', cont
    includedCreditsPerUser: 15,
    rateLimits: { requestsPerMinute: 20_000, tokensPerMinute: 50_000_000, concurrentRequests:
    features: {
      modelAccess: 'all', maxModelsPerMonth: null, maxRequestsPerDay: null,
      maxTokensPerRequest: 2_000_000, priorityQueue: true, apiAccess: true,
      exportFormats: ['txt', 'md', 'pdf', 'docx', 'html', 'json', 'csv', 'xml'],
      supportLevel: 'dedicated', dataRetention: 'custom', watermarkedOutputs: false,
      conversationHistory: true, customInstructions: true, sharedCreditPool: true,
      teamWorkspaces: true, sharedConversations: true, teamAdminDashboard: true,
      usageAnalytics: true, roleBasedAccess: true, inviteManagement: true,
      ssoIntegration: true, scimProvisioning: true, auditLogs: true, dataExport: true,
      customBranding: true, dedicatedAccountManager: true,
      apiRateLimitCustomization: true, webhooks: true, ipWhitelisting: true,
      slaGuarantee: true, uptimeGuarantee: '99.99%', customModelFineTuning: true,
      dedicatedInfrastructure: true, multiRegionDeployment: true, onPremiseDeployment: true,
    },
    availableAddOns: [], // Compliance included
    isPublic: true,
    sortOrder: 6,
    badge: 'Full Compliance Included',
  },
};

// ============================================================================
// HELPER FUNCTIONS
// ============================================================================

export function getTier(tierId: SubscriptionTierId): SubscriptionTier {
  return SUBSCRIPTION_TIERS[tierId];
}

export function canUpgrade(fromTier: SubscriptionTierId, toTier: SubscriptionTierId): boolean
  return SUBSCRIPTION_TIERS[toTier].sortOrder > SUBSCRIPTION_TIERS[fromTier].sortOrder;
}

export function getAnnualSavings(tier: SubscriptionTier): number {
```

```
  if (!tier.pricing.monthly || !tier.pricing.annual) return 0;
  return (tier.pricing.monthly * 12) - tier.pricing.annual;
}
```

43.3 CREDITS SYSTEM

packages/shared/src/billing/credits.ts

```
/**
 * RADIANT Credits System
 * Prepaid AI usage currency
 *
 * @version 4.13.0
 * @module @radiant/shared/billing
 */

// ============================================================================
// CREDIT TYPES
// ============================================================================

export interface CreditPool {
  id: string;
  type: CreditPoolType;
  ownerId: string;
  organizationId?: string;
  tenantId: string;

  balance: {
    available: number;
    reserved: number;
    total: number;
  };

  monthlyIncluded: {
    total: number;
    used: number;
    remaining: number;
    resetsAt: string;
  };

  purchased: {
    total: number;
    remaining: number;
    lastPurchaseAt?: string;
  };

  bonus: {
```

```
    total: number;
    remaining: number;
  };

  members: CreditPoolMember[];
  settings: CreditPoolSettings;

  createdAt: string;
  updatedAt: string;
}

export type CreditPoolType = 'individual' | 'family' | 'team' | 'organization' | 'enterprise';

export interface CreditPoolMember {
  userId: string;
  email: string;
  displayName: string;
  role: 'owner' | 'admin' | 'member' | 'restricted';

  limits?: {
    dailyCredits?: number;
    monthlyCredits?: number;
    maxCostPerRequest?: number;
  };

  permissions: CreditPoolPermissions;
  status: 'active' | 'invited' | 'suspended';
  joinedAt: string;
  lastActiveAt?: string;

  usage: {
    today: number;
    thisWeek: number;
    thisMonth: number;
    allTime: number;
  };
}

export interface CreditPoolPermissions {
  canPurchaseCredits: boolean;
  canViewPoolBalance: boolean;
  canViewOtherUsage: boolean;
  canInviteMembers: boolean;
  canRemoveMembers: boolean;
  canModifySettings: boolean;
}

export interface CreditPoolSettings {
```

```
  autoPurchase: {
    enabled: boolean;
    thresholdCredits: number;
    purchaseAmount: number;
    maxMonthlyAutoPurchase?: number;
    paymentMethodId?: string;
  };

  notifications: {
    lowBalanceThreshold: number;
    lowBalanceRecipients: string[];
    usageSummaryFrequency: 'daily' | 'weekly' | 'monthly';
    unusualUsageAlerts: boolean;
  };

  accessControl: {
    requireApprovalAbove?: number;
    blockedModels?: string[];
    allowedModels?: string[];
  };

  familySettings?: {
    parentalControlsEnabled: boolean;
    contentFiltering: 'strict' | 'moderate' | 'off';
    underageMembers: string[];
  };
}

// ============================================================================
// CREDIT TRANSACTIONS
// ============================================================================

export type CreditTransactionType =
  | 'purchase' | 'subscription_grant' | 'bonus_grant' | 'consumption'
  | 'refund' | 'adjustment' | 'transfer_in' | 'transfer_out' | 'expiration';

export interface CreditTransaction {
  id: string;
  poolId: string;
  userId?: string;
  transactionType: CreditTransactionType;

  amount: number;
  balanceAfter: number;

  sourceType?: 'included' | 'purchased' | 'bonus';
  sourceId?: string;
```

```
  modelId?: string;
  requestId?: string;
  inputTokens?: number;
  outputTokens?: number;

  paymentIntentId?: string;
  paymentAmountCents?: number;
  paymentCurrency?: string;

  description?: string;
  metadata?: Record<string, any>;

  createdAt: string;
}

// ============================================================================
// CREDIT PRICING
// ============================================================================

export const CREDIT_PRICING = {
  basePrice: 10.00, // $10 = 1 Credit

  volumeDiscounts: [
    { minCredits: 1, discount: 0, bonus: 0 },
    { minCredits: 5, discount: 0, bonus: 0 },
    { minCredits: 10, discount: 0.05, bonus: 0.05 },   // 5% off, 5% bonus
    { minCredits: 25, discount: 0.10, bonus: 0.10 },   // 10% off, 10% bonus
    { minCredits: 50, discount: 0.15, bonus: 0.15 },   // 15% off, 15% bonus
    { minCredits: 100, discount: 0.20, bonus: 0.20 },  // 20% off, 20% bonus
  ],

  consumption: {
    text: {
      inputTokensPer1Credit: 1_000_000,
      outputTokensPer1Credit: 250_000,
    },
    image: {
      standardGenerationCredits: 0.02,
      hdGenerationCredits: 0.04,
    },
    audio: {
      transcriptionMinuteCredits: 0.01,
      ttsCharacterCredits: 0.00001,
    },
  },
};

export function calculatePurchasePrice(credits: number): {
```

```
  basePrice: number;
  discount: number;
  bonusCredits: number;
  finalPrice: number;
  totalCredits: number;
} {
  const tier = [...CREDIT_PRICING.volumeDiscounts]
    .reverse()
    .find(t => credits >= t.minCredits) || CREDIT_PRICING.volumeDiscounts[0];

  const basePrice = credits * CREDIT_PRICING.basePrice;
  const discount = basePrice * tier.discount;
  const bonusCredits = credits * tier.bonus;

  return {
    basePrice,
    discount,
    bonusCredits,
    finalPrice: basePrice - discount,
    totalCredits: credits + bonusCredits,
  };
}
```

43.4 DATABASE SCHEMA

packages/infrastructure/migrations/043_billing_system.sql

```
-- ============================================================================
-- RADIANT v4.13.0 - Billing & Credits Schema
-- Migration: 043_billing_system.sql
-- ============================================================================

-- Subscription Tiers
CREATE TABLE subscription_tiers (
    id VARCHAR(50) PRIMARY KEY,
    display_name VARCHAR(100) NOT NULL,
    description TEXT,

    price_monthly DECIMAL(10,2),
    price_annual DECIMAL(10,2),
    price_per_user BOOLEAN DEFAULT FALSE,
    min_users INTEGER DEFAULT 1,
    max_users INTEGER,
    currency VARCHAR(3) DEFAULT 'USD',

    included_credits_per_user DECIMAL(10,2) NOT NULL DEFAULT 0,
```

```
    requests_per_minute INTEGER NOT NULL,
    tokens_per_minute BIGINT NOT NULL,
    concurrent_requests INTEGER NOT NULL,

    features JSONB NOT NULL DEFAULT '{}',
    available_add_ons TEXT[] DEFAULT '{}',

    is_active BOOLEAN DEFAULT TRUE,
    is_public BOOLEAN DEFAULT TRUE,
    sort_order INTEGER DEFAULT 0,
    badge VARCHAR(50),

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Subscription Add-Ons
CREATE TABLE subscription_add_ons (
    id VARCHAR(50) PRIMARY KEY,
    display_name VARCHAR(100) NOT NULL,
    description TEXT,

    price_monthly_cents INTEGER NOT NULL,
    price_annual_cents INTEGER,
    price_per_user BOOLEAN DEFAULT FALSE,

    available_for_tiers TEXT[] NOT NULL,
    included_in_tiers TEXT[] DEFAULT '{}',

    features JSONB NOT NULL DEFAULT '{}',

    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Credit Pools
CREATE TABLE credit_pools (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_type VARCHAR(20) NOT NULL CHECK (pool_type IN ('individual', 'family', 'team', 'organi
    owner_user_id UUID NOT NULL REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),

    balance_available DECIMAL(12,4) NOT NULL DEFAULT 0,
    balance_reserved DECIMAL(12,4) NOT NULL DEFAULT 0,

    monthly_included_total DECIMAL(12,4) NOT NULL DEFAULT 0,
```

```
    monthly_included_used DECIMAL(12,4) NOT NULL DEFAULT 0,
    monthly_resets_at TIMESTAMPTZ,

    purchased_total DECIMAL(12,4) NOT NULL DEFAULT 0,
    purchased_remaining DECIMAL(12,4) NOT NULL DEFAULT 0,
    last_purchase_at TIMESTAMPTZ,

    bonus_total DECIMAL(12,4) NOT NULL DEFAULT 0,
    bonus_remaining DECIMAL(12,4) NOT NULL DEFAULT 0,

    settings JSONB NOT NULL DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_pools_owner ON credit_pools(owner_user_id);
CREATE INDEX idx_credit_pools_org ON credit_pools(organization_id);
CREATE INDEX idx_credit_pools_tenant ON credit_pools(tenant_id);

-- Credit Pool Members
CREATE TABLE credit_pool_members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id) ON DELETE CASCADE,
    user_id UUID NOT NULL REFERENCES users(id),

    role VARCHAR(20) NOT NULL DEFAULT 'member' CHECK (role IN ('owner', 'admin', 'member', 'res
    permissions JSONB NOT NULL DEFAULT '{}',

    daily_credit_limit DECIMAL(10,4),
    monthly_credit_limit DECIMAL(10,4),
    max_cost_per_request DECIMAL(10,4),

    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'invited', 'suspe
    invited_by UUID REFERENCES users(id),
    invited_at TIMESTAMPTZ,
    joined_at TIMESTAMPTZ,
    suspended_at TIMESTAMPTZ,
    suspended_reason TEXT,

    usage_today DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_this_week DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_this_month DECIMAL(12,4) NOT NULL DEFAULT 0,
    usage_all_time DECIMAL(12,4) NOT NULL DEFAULT 0,
    last_usage_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW(),
```

```
    UNIQUE(pool_id, user_id)
);

CREATE INDEX idx_pool_members_user ON credit_pool_members(user_id);
CREATE INDEX idx_pool_members_pool ON credit_pool_members(pool_id);

-- Credit Transactions
CREATE TABLE credit_transactions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id),
    user_id UUID REFERENCES users(id),

    transaction_type VARCHAR(30) NOT NULL CHECK (transaction_type IN (
        'purchase', 'subscription_grant', 'bonus_grant', 'consumption',
        'refund', 'adjustment', 'transfer_in', 'transfer_out', 'expiration'
    )),

    amount DECIMAL(12,4) NOT NULL,
    balance_after DECIMAL(12,4) NOT NULL,

    source_type VARCHAR(30),
    source_id VARCHAR(100),

    model_id VARCHAR(100),
    request_id UUID,
    input_tokens INTEGER,
    output_tokens INTEGER,

    payment_intent_id VARCHAR(100),
    payment_amount_cents INTEGER,
    payment_currency VARCHAR(3),

    description TEXT,
    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_transactions_pool ON credit_transactions(pool_id);
CREATE INDEX idx_credit_transactions_user ON credit_transactions(user_id);
CREATE INDEX idx_credit_transactions_type ON credit_transactions(transaction_type);
CREATE INDEX idx_credit_transactions_created ON credit_transactions(created_at);

-- Credit Purchases
CREATE TABLE credit_purchases (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id),
```

```
    user_id UUID NOT NULL REFERENCES users(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),

    credits_amount DECIMAL(10,4) NOT NULL,
    bonus_credits DECIMAL(10,4) NOT NULL DEFAULT 0,
    total_credits DECIMAL(10,4) NOT NULL,

    payment_amount_cents INTEGER NOT NULL,
    payment_currency VARCHAR(3) NOT NULL DEFAULT 'USD',
    payment_method VARCHAR(50),

    stripe_payment_intent_id VARCHAR(100),
    stripe_charge_id VARCHAR(100),
    stripe_invoice_id VARCHAR(100),
    stripe_receipt_url TEXT,

    status VARCHAR(20) NOT NULL DEFAULT 'pending' CHECK (status IN (
        'pending', 'processing', 'completed', 'failed', 'refunded', 'partially_refunded'
    )),
    completed_at TIMESTAMPTZ,
    failed_at TIMESTAMPTZ,
    failure_reason TEXT,

    is_auto_purchase BOOLEAN DEFAULT FALSE,
    auto_purchase_trigger VARCHAR(50),

    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_purchases_pool ON credit_purchases(pool_id);
CREATE INDEX idx_credit_purchases_user ON credit_purchases(user_id);
CREATE INDEX idx_credit_purchases_status ON credit_purchases(status);
CREATE INDEX idx_credit_purchases_stripe ON credit_purchases(stripe_payment_intent_id);

-- Subscriptions
CREATE TABLE subscriptions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    user_id UUID NOT NULL REFERENCES users(id),
    organization_id UUID REFERENCES organizations(id),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    credit_pool_id UUID REFERENCES credit_pools(id),

    tier_id VARCHAR(50) NOT NULL REFERENCES subscription_tiers(id),
    billing_cycle VARCHAR(10) NOT NULL CHECK (billing_cycle IN ('monthly', 'annual')),
```

```
    seat_count INTEGER NOT NULL DEFAULT 1,
    add_ons JSONB NOT NULL DEFAULT '[]',

    price_per_unit_cents INTEGER NOT NULL,
    total_price_cents INTEGER NOT NULL,
    currency VARCHAR(3) NOT NULL DEFAULT 'USD',

    stripe_subscription_id VARCHAR(100),
    stripe_customer_id VARCHAR(100) NOT NULL,
    stripe_price_id VARCHAR(100),

    current_period_start TIMESTAMPTZ NOT NULL,
    current_period_end TIMESTAMPTZ NOT NULL,
    trial_start TIMESTAMPTZ,
    trial_end TIMESTAMPTZ,

    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN (
        'trialing', 'active', 'past_due', 'canceled', 'unpaid', 'paused'
    )),
    cancel_at_period_end BOOLEAN DEFAULT FALSE,
    canceled_at TIMESTAMPTZ,
    cancellation_reason TEXT,

    metadata JSONB DEFAULT '{}',

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_subscriptions_user ON subscriptions(user_id);
CREATE INDEX idx_subscriptions_org ON subscriptions(organization_id);
CREATE INDEX idx_subscriptions_tenant ON subscriptions(tenant_id);
CREATE INDEX idx_subscriptions_stripe ON subscriptions(stripe_subscription_id);
CREATE INDEX idx_subscriptions_status ON subscriptions(status);

-- Auto-Purchase Settings
CREATE TABLE auto_purchase_settings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    pool_id UUID NOT NULL REFERENCES credit_pools(id) UNIQUE,

    enabled BOOLEAN NOT NULL DEFAULT FALSE,
    threshold_credits DECIMAL(10,4) NOT NULL DEFAULT 1.0,
    purchase_credits DECIMAL(10,4) NOT NULL DEFAULT 10.0,
    max_monthly_auto_purchase DECIMAL(10,4),

    stripe_payment_method_id VARCHAR(100),
```

```
    auto_purchases_this_month INTEGER NOT NULL DEFAULT 0,
    auto_purchase_spend_this_month DECIMAL(10,2) NOT NULL DEFAULT 0,
    last_auto_purchase_at TIMESTAMPTZ,
    month_reset_at TIMESTAMPTZ,

    notify_on_auto_purchase BOOLEAN DEFAULT TRUE,
    notification_emails TEXT[],

    created_at TIMESTAMPTZ DEFAULT NOW(),
    updated_at TIMESTAMPTZ DEFAULT NOW()
);

-- Credit Usage (for analytics)
CREATE TABLE credit_usage (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

    pool_id UUID NOT NULL REFERENCES credit_pools(id),
    user_id UUID NOT NULL REFERENCES users(id),
    transaction_id UUID REFERENCES credit_transactions(id),

    request_id UUID NOT NULL,
    model_id VARCHAR(100) NOT NULL,
    provider_id VARCHAR(50) NOT NULL,

    input_tokens INTEGER NOT NULL DEFAULT 0,
    output_tokens INTEGER NOT NULL DEFAULT 0,
    cached_tokens INTEGER NOT NULL DEFAULT 0,

    credits_consumed DECIMAL(12,6) NOT NULL,
    credit_rate_multiplier DECIMAL(6,4) NOT NULL DEFAULT 1.0,

    credits_from_included DECIMAL(12,6) NOT NULL DEFAULT 0,
    credits_from_purchased DECIMAL(12,6) NOT NULL DEFAULT 0,
    credits_from_bonus DECIMAL(12,6) NOT NULL DEFAULT 0,

    request_type VARCHAR(50),
    latency_ms INTEGER,

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_credit_usage_pool ON credit_usage(pool_id);
CREATE INDEX idx_credit_usage_user ON credit_usage(user_id);
CREATE INDEX idx_credit_usage_model ON credit_usage(model_id);
CREATE INDEX idx_credit_usage_created ON credit_usage(created_at);

-- Billing Events
CREATE TABLE billing_events (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    event_type VARCHAR(50) NOT NULL,
    stripe_event_id VARCHAR(100),

    pool_id UUID REFERENCES credit_pools(id),
    user_id UUID REFERENCES users(id),
    subscription_id UUID REFERENCES subscriptions(id),
    purchase_id UUID REFERENCES credit_purchases(id),

    data JSONB NOT NULL DEFAULT '{}',

    processed BOOLEAN DEFAULT FALSE,
    processed_at TIMESTAMPTZ,
    error TEXT,
    retry_count INTEGER DEFAULT 0,

    created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE INDEX idx_billing_events_type ON billing_events(event_type);
CREATE INDEX idx_billing_events_processed ON billing_events(processed) WHERE processed = FALSE
CREATE INDEX idx_billing_events_stripe ON billing_events(stripe_event_id);

-- Row Level Security
ALTER TABLE credit_pools ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_pool_members ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_transactions ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_purchases ENABLE ROW LEVEL SECURITY;
ALTER TABLE subscriptions ENABLE ROW LEVEL SECURITY;
ALTER TABLE credit_usage ENABLE ROW LEVEL SECURITY;

CREATE POLICY credit_pools_tenant_isolation ON credit_pools
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY credit_pool_members_access ON credit_pool_members FOR SELECT
    USING (
        user_id = current_setting('app.current_user_id')::UUID OR
        pool_id IN (
            SELECT pool_id FROM credit_pool_members
            WHERE user_id = current_setting('app.current_user_id')::UUID
            AND role IN ('owner', 'admin')
        )
    );
```

43.5 CREDITS SERVICE

packages/functions/src/services/credits.ts

```
/**
 * Credits Service
 * Manages credit consumption, purchases, and balances
 *
 * @version 4.13.0
 */

import { Pool } from 'pg';
import Stripe from 'stripe';
import { CREDIT_PRICING, calculatePurchasePrice } from '@radiant/shared/billing/credits';

export class CreditsService {
  constructor(
    private pool: Pool,
    private stripe: Stripe
  ) {}

  async getBalance(userId: string): Promise<{
    available: number;
    reserved: number;
    includedRemaining: number;
    purchasedRemaining: number;
    bonusRemaining: number;
  }> {
    const result = await this.pool.query(`
      SELECT
        cp.balance_available,
        cp.balance_reserved,
        cp.monthly_included_total - cp.monthly_included_used as included_remaining,
        cp.purchased_remaining,
        cp.bonus_remaining
      FROM credit_pools cp
      JOIN credit_pool_members cpm ON cpm.pool_id = cp.id
      WHERE cpm.user_id = $1 AND cpm.status = 'active'
    `, [userId]);

    if (result.rows.length === 0) {
      return { available: 0, reserved: 0, includedRemaining: 0, purchasedRemaining: 0, bonusRe
    }

    const row = result.rows[0];
    return {
      available: parseFloat(row.balance_available),
      reserved: parseFloat(row.balance_reserved),
```

```
    includedRemaining: parseFloat(row.included_remaining),
    purchasedRemaining: parseFloat(row.purchased_remaining),
    bonusRemaining: parseFloat(row.bonus_remaining),
  };
}

async reserveCredits(userId: string, amount: number): Promise<boolean> {
  const result = await this.pool.query(`
    UPDATE credit_pools cp
    SET
      balance_available = balance_available - $2,
      balance_reserved = balance_reserved + $2,
      updated_at = NOW()
    FROM credit_pool_members cpm
    WHERE cpm.pool_id = cp.id
      AND cpm.user_id = $1
      AND cpm.status = 'active'
      AND cp.balance_available >= $2
    RETURNING cp.id
  `, [userId, amount]);

  return result.rowCount > 0;
}

async consumeCredits(
  userId: string,
  modelId: string,
  inputTokens: number,
  outputTokens: number,
  requestId: string
): Promise<{ consumed: number; remaining: number }> {
  const creditCost = this.calculateCreditCost(modelId, inputTokens, outputTokens);

  const client = await this.pool.connect();
  try {
    await client.query('BEGIN');

    // Consume from reserved first, then available
    const consumeResult = await client.query(`
      UPDATE credit_pools cp
      SET
        balance_reserved = GREATEST(0, balance_reserved - $2),
        balance_available = CASE
          WHEN balance_reserved >= $2 THEN balance_available
          ELSE balance_available - ($2 - balance_reserved)
        END,
        updated_at = NOW()
      FROM credit_pool_members cpm
```

```
  WHERE cpm.pool_id = cp.id
    AND cpm.user_id = $1
    AND cpm.status = 'active'
  RETURNING cp.id, cp.balance_available
`, [userId, creditCost]);

if (consumeResult.rows.length === 0) {
  throw new Error('No active credit pool found');
}

// Record transaction
await client.query(`
  INSERT INTO credit_transactions (
    pool_id, user_id, transaction_type, amount, balance_after,
    model_id, request_id, input_tokens, output_tokens
  ) VALUES ($1, $2, 'consumption', $3, $4, $5, $6, $7, $8)
`, [
  consumeResult.rows[0].id, userId, -creditCost,
  consumeResult.rows[0].balance_available, modelId, requestId,
  inputTokens, outputTokens
]);

// Record usage for analytics
await client.query(`
  INSERT INTO credit_usage (
    pool_id, user_id, request_id, model_id, provider_id,
    input_tokens, output_tokens, credits_consumed
  ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)
`, [
  consumeResult.rows[0].id, userId, requestId, modelId, 'auto',
  inputTokens, outputTokens, creditCost
]);

// Update member usage stats
await client.query(`
  UPDATE credit_pool_members
  SET
    usage_today = usage_today + $2,
    usage_this_month = usage_this_month + $2,
    usage_all_time = usage_all_time + $2,
    last_usage_at = NOW(),
    updated_at = NOW()
  WHERE user_id = $1
`, [userId, creditCost]);

await client.query('COMMIT');

return {
```

```
      consumed: creditCost,
      remaining: parseFloat(consumeResult.rows[0].balance_available),
    };
  } catch (error) {
    await client.query('ROLLBACK');
    throw error;
  } finally {
    client.release();
  }
}

async purchaseCredits(
  userId: string,
  poolId: string,
  credits: number,
  paymentMethodId: string
): Promise<{ purchaseId: string; totalCredits: number }> {
  const pricing = calculatePurchasePrice(credits);

  // Create Stripe payment intent
  const paymentIntent = await this.stripe.paymentIntents.create({
    amount: Math.round(pricing.finalPrice * 100),
    currency: 'usd',
    payment_method: paymentMethodId,
    confirm: true,
    metadata: {
      poolId,
      userId,
      credits: credits.toString(),
      bonusCredits: pricing.bonusCredits.toString(),
    },
  });

  // Record purchase
  const result = await this.pool.query(`
    INSERT INTO credit_purchases (
      pool_id, user_id, tenant_id,
      credits_amount, bonus_credits, total_credits,
      payment_amount_cents, stripe_payment_intent_id,
      status, completed_at
    )
    SELECT
      $1, $2, cp.tenant_id,
      $3, $4, $5, $6, $7, 'completed', NOW()
    FROM credit_pools cp WHERE cp.id = $1
    RETURNING id
  `, [
    poolId, userId,
```

```
      credits, pricing.bonusCredits, pricing.totalCredits,
      Math.round(pricing.finalPrice * 100), paymentIntent.id
    ]);

    // Add credits to pool
    await this.pool.query(`
      UPDATE credit_pools
      SET
        purchased_total = purchased_total + $2,
        purchased_remaining = purchased_remaining + $2,
        bonus_total = bonus_total + $3,
        bonus_remaining = bonus_remaining + $3,
        balance_available = balance_available + $2 + $3,
        last_purchase_at = NOW(),
        updated_at = NOW()
      WHERE id = $1
    `, [poolId, credits, pricing.bonusCredits]);

    return {
      purchaseId: result.rows[0].id,
      totalCredits: pricing.totalCredits,
    };
  }

  private calculateCreditCost(modelId: string, inputTokens: number, outputTokens: number): numb
    const inputCredits = inputTokens / CREDIT_PRICING.consumption.text.inputTokensPer1Credit;
    const outputCredits = outputTokens / CREDIT_PRICING.consumption.text.outputTokensPer1Credi
    return inputCredits + outputCredits;
  }
}
```
