TRADEWISER COMPLETE REBUILD - WORKING AGRICULTURAL FINTECH PLATFORM

CONTEXT

The previous implementation has significant gaps. This prompt focuses on building a WORKING end-to-end agricultural fintech platform that functions like Zerodha for commodities - where users can deposit, get receipts, take loans, and manage their portfolio seamlessly.

OBJECTIVE

Build a complete, working agricultural fintech platform with functioning deposit-to-loan workflows, real-time portfolio management, and banking-grade UX.

PHASE 1: CORE DATA FLOW - MAKE IT WORK

1.1 Fix Database Schema & Data Persistence

Database Schema Fixes

```
-- Ensure proper database structure
CREATE TABLE IF NOT EXISTS commodities (
    id SERIAL PRIMARY KEY,
    owner_id INTEGER REFERENCES users(id),
    name VARCHAR(255) NOT NULL,
    category VARCHAR(100) NOT NULL,
    quantity DECIMAL(10,2) NOT NULL,
    unit VARCHAR(20) NOT NULL DEFAULT 'MT',
    quality_grade VARCHAR(50),
    market_value DECIMAL(12,2),
    warehouse_id INTEGER,
    status VARCHAR(50) DEFAULT 'active',
    metadata JSONB,
    created_at TIMESTAMP DEFAULT NOW(),
    updated_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS warehouse_receipts (
    id SERIAL PRIMARY KEY,
    receipt_number VARCHAR(100) UNIQUE NOT NULL,
    commodity_id INTEGER REFERENCES commodities(id),
    owner_id INTEGER REFERENCES users(id),
    quantity DECIMAL(10,2) NOT NULL,
    valuation DECIMAL(12,2) NOT NULL,
    quality_grade VARCHAR(50),
    status VARCHAR(50) DEFAULT 'active',
    available_for_collateral BOOLEAN DEFAULT true,
```

```
    collateral_used DECIMAL(12,2) DEFAULT 0,
    blockchain_hash VARCHAR(255),
    created_at TIMESTAMP DEFAULT NOW()
);

CREATE TABLE IF NOT EXISTS loans (
    id SERIAL PRIMARY KEY,
    borrower_id INTEGER REFERENCES users(id),
    collateral_receipt_id INTEGER REFERENCES warehouse_receipts(id),
    amount DECIMAL(12,2) NOT NULL,
    interest_rate DECIMAL(5,2) DEFAULT 12.0,
    duration_months INTEGER DEFAULT 12,
    status VARCHAR(50) DEFAULT 'active',
    disbursed_at TIMESTAMP DEFAULT NOW(),
    due_date TIMESTAMP,
    monthly_emi DECIMAL(10,2),
    outstanding_amount DECIMAL(12,2),
    created_at TIMESTAMP DEFAULT NOW()
);

-- Create indexes for performance
CREATE INDEX idx_commodities_owner ON commodities(owner_id);
CREATE INDEX idx_receipts_owner ON warehouse_receipts(owner_id);
CREATE INDEX idx_loans_borrower ON loans(borrower_id);
```

Core API Endpoints - WORKING IMPLEMENTATION

```
// DEPOSIT API - ACTUALLY WORKS
app.post('/api/deposits', async (req, res) => {
  try {
    const { commodityName, commodityType, quantity, unit, qualityParams, location } = req

    // Calculate market value (mock pricing for now)
    const basePrice = getCommodityBasePrice(commodityName);
    const marketValue = basePrice * parseFloat(quantity);

    // Create commodity entry
    const commodity = await db.insert(commodities).values({
      ownerId: req.user.id,
      name: commodityName,
      category: commodityType,
      quantity: parseFloat(quantity),
      unit: unit || 'MT',
      marketValue: marketValue,
      qualityGrade: 'Pending Assessment',
      status: 'deposited',
      metadata: {
        qualityParams,
        location,
        depositedAt: new Date()
      }
    }).returning();

    // IMMEDIATELY create warehouse receipt (no complex tracking for now)
    const receiptNumber = `TW${Date.now()}-${Math.random().toString(36).substr(2, 6)}`;
```

```
    const receipt = await db.insert(warehouseReceipts).values({
      receiptNumber,
      commodityId: commodity[0].id,
      ownerId: req.user.id,
      quantity: parseFloat(quantity),
      valuation: marketValue,
      qualityGrade: 'Grade A', // Mock quality for demo
      status: 'active',
      availableForCollateral: true,
      blockchainHash: `BC-${Date.now()}`
    }).returning();

    res.json({
      success: true,
      data: {
        commodity: commodity[0],
        receipt: receipt[0],
        message: 'Commodity deposited and receipt generated successfully!'
      }
    });
  } catch (error) {
    console.error('Deposit error:', error);
    res.status(500).json({ success: false, error: error.message });
  }
});

// PORTFOLIO API - WORKING DATA
app.get('/api/portfolio', async (req, res) => {
  try {
    const receipts = await db.select().from(warehouseReceipts)
      .where(eq(warehouseReceipts.ownerId, req.user.id));

    const commodities = await db.select().from(commodities)
      .where(eq(commodities.ownerId, req.user.id));

    const totalValue = receipts.reduce((sum, receipt) => sum + (receipt.valuation || 0),
    const availableCredit = totalValue * 0.8; // 80% LTV

    res.json({
      success: true,
      data: {
        totalValue,
        receiptsCount: receipts.length,
        availableCredit,
        receipts,
        commodities
      }
    });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

// ELIGIBLE RECEIPTS FOR LOANS - WORKING
app.get('/api/receipts/eligible-for-loans', async (req, res) => {
  try {
```

```
    const receipts = await db.select().from(warehouseReceipts)
      .where(
        and(
          eq(warehouseReceipts.ownerId, req.user.id),
          eq(warehouseReceipts.status, 'active'),
          eq(warehouseReceipts.availableForCollateral, true)
        )
      );

    // Calculate available loan amount for each receipt
    const eligibleReceipts = receipts.map(receipt => ({
      ...receipt,
      availableLoanAmount: (receipt.valuation - (receipt.collateralUsed || 0)) * 0.8
    })).filter(receipt => receipt.availableLoanAmount > 0);

    res.json({ success: true, data: eligibleReceipts });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

// LOAN APPLICATION - WORKING
app.post('/api/loans', async (req, res) => {
  try {
    const { receiptId, amount, durationMonths } = req.body;

    const receipt = await db.select().from(warehouseReceipts)
      .where(eq(warehouseReceipts.id, receiptId))
      .limit(1);

    if (!receipt[0]) {
      return res.status(400).json({ success: false, error: 'Receipt not found' });
    }

    const maxLoanAmount = (receipt[0].valuation - (receipt[0].collateralUsed || 0)) * 0.8

    if (amount > maxLoanAmount) {
      return res.status(400).json({
        success: false,
        error: `Maximum available: ₹${maxLoanAmount.toLocaleString()}`
      });
    }

    // Calculate EMI
    const monthlyRate = 0.12 / 12; // 12% annual
    const emi = (amount * monthlyRate * Math.pow(1 + monthlyRate, durationMonths)) /
                (Math.pow(1 + monthlyRate, durationMonths) - 1);

    const loan = await db.insert(loans).values({
      borrowerId: req.user.id,
      collateralReceiptId: receiptId,
      amount: parseFloat(amount),
      interestRate: 12.0,
      durationMonths: parseInt(durationMonths),
      monthlyEmi: emi,
      outstandingAmount: parseFloat(amount),
```

```
      status: 'active',
      dueDate: new Date(Date.now() + durationMonths * 30 * 24 * 60 * 60 * 1000)
    }).returning();

    // Update collateral usage
    await db.update(warehouseReceipts)
      .set({ collateralUsed: (receipt[0].collateralUsed || 0) + parseFloat(amount) })
      .where(eq(warehouseReceipts.id, receiptId));

    res.json({ success: true, data: loan[0] });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

// Helper function for commodity pricing
function getCommodityBasePrice(commodity) {
  const prices = {
    'Wheat': 2500,
    'Rice': 3000,
    'Maize': 2000,
    'Soybean': 4500,
    'Cotton': 6000,
    'Sugarcane': 300
  };
  return prices[commodity] || 2500; // Default price per MT
}
```

PHASE 2: STREAMLINED UX - BANKING GRADE EXPERIENCE

2.1 Single-Page Deposit Form - ACTUALLY SIMPLE

```
const StreamlinedDepositForm = () => {
  const [formData, setFormData] = useState({
    commodityName: '',
    commodityType: '',
    quantity: '',
    unit: 'MT'
  });
  const [estimatedValue, setEstimatedValue] = useState(0);
  const [isSubmitting, setIsSubmitting] = useState(false);

  // Real-time value calculation
  useEffect(() => {
    if (formData.commodityName && formData.quantity) {
      const basePrice = getCommodityPrice(formData.commodityName);
      setEstimatedValue(basePrice * parseFloat(formData.quantity || 0));
    }
  }, [formData.commodityName, formData.quantity]);

  const handleSubmit = async (e) => {
```

```
    e.preventDefault();
    setIsSubmitting(true);

    try {
      const response = await fetch('/api/deposits', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify(formData)
      });

      const result = await response.json();

      if (result.success) {
        toast.success(`✅ Deposit successful! Receipt ${result.data.receipt.receiptNumber
        // Refresh portfolio data
        queryClient.invalidateQueries(['portfolio']);
        queryClient.invalidateQueries(['eligible-receipts']);
        // Redirect to dashboard
        setTimeout(() => router.push('/dashboard'), 2000);
      } else {
        toast.error(result.error || 'Deposit failed');
      }
    } catch (error) {
      toast.error('Network error. Please try again.');
    } finally {
      setIsSubmitting(false);
    }
  };

  return (
    <div className="max-w-2xl mx-auto p-6">
      <div className="bg-white rounded-xl shadow-sm border p-8">
        <h1 className="text-3xl font-bold text-gray-900 mb-8">Deposit Commodity</h1>

        <form onSubmit={handleSubmit} className="space-y-6">
          {/* Commodity Selection */}
          <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                Commodity Name *
              </label>
              <CommoditySelector
                value={formData.commodityName}
                onChange={(value) => setFormData(prev => ({
                  ...prev,
                  commodityName: value,
                  commodityType: getCommodityCategory(value)
                }))}
                required
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                Category
```

```
              </label>
              <Input
                value={formData.commodityType}
                disabled
                className="bg-gray-50"
              />
            </div>
          </div>

          {/* Quantity */}
          <div className="grid grid-cols-2 gap-6">
            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                Quantity *
              </label>
              <Input
                type="number"
                step="0.01"
                placeholder="Enter quantity"
                value={formData.quantity}
                onChange={(e) => setFormData(prev => ({ ...prev, quantity: e.target.value
                required
              />
            </div>

            <div>
              <label className="block text-sm font-medium text-gray-700 mb-2">
                Unit
              </label>
              <Select
                value={formData.unit}
                onValueChange={(value) => setFormData(prev => ({ ...prev, unit: value }))
              >
                <SelectTrigger>
                  <SelectValue />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="MT">Metric Tons (MT)</SelectItem>
                  <SelectItem value="quintal">Quintals</SelectItem>
                  <SelectItem value="kg">Kilograms</SelectItem>
                </SelectContent>
              </Select>
            </div>
          </div>

          {/* Estimated Value */}
          {estimatedValue > 0 && (
            <div className="bg-green-50 border border-green-200 rounded-lg p-6">
              <h3 className="font-semibold text-green-900 mb-4">Estimated Value</h3>
              <div className="grid grid-cols-2 gap-4 text-sm">
                <div className="flex justify-between">
                  <span className="text-green-700">Market Value:</span>
                  <span className="font-medium">₹{estimatedValue.toLocaleString()}</span>
                </div>
                <div className="flex justify-between">
                  <span className="text-green-700">Available for Loan:</span>
```

```
                  <span className="font-medium">₹{(estimatedValue * 0.8).toLocaleString()
                </div>
              </div>
            </div>
          )}

          {/* Submit Button */}
          <Button
            type="submit"
            size="lg"
            className="w-full bg-green-600 hover:bg-green-700"
            disabled={isSubmitting || !formData.commodityName || !formData.quantity}
          >
            {isSubmitting ? (
              <div className="flex items-center gap-2">
                <Loader2 className="w-5 h-5 animate-spin" />
                Creating Receipt...
              </div>
            ) : (
              'Deposit & Generate Receipt'
            )}
          </Button>
        </form>
      </div>
    </div>
  );
};
```

2.2 Portfolio Dashboard - Like Zerodha

```
const PortfolioDashboard = () => {
  const { data: portfolio, isLoading } = useQuery(['portfolio'],
    () => fetch('/api/portfolio', { credentials: 'include' }).then(r => r.json())
  );

  const { data: eligibleReceipts } = useQuery(['eligible-receipts'],
    () => fetch('/api/receipts/eligible-for-loans', { credentials: 'include' }).then(r =>
  );

  const { data: activeLoans } = useQuery(['active-loans'],
    () => fetch('/api/loans', { credentials: 'include' }).then(r => r.json())
  );

  if (isLoading) return <div className="flex justify-center py-12"><Loader2 className="w-

  const portfolioData = portfolio?.data || {};

  return (
    <div className="max-w-7xl mx-auto p-6 space-y-8">
      {/* Header */}
      <div>
        <h1 className="text-3xl font-bold text-gray-900">Portfolio Dashboard</h1>
        <p className="text-gray-600 mt-2">Manage your commodity holdings and financing</p
      </div>
```

```
      {/* Portfolio Overview Cards */}
      <div className="grid grid-cols-1 md:grid-cols-4 gap-6">
        <div className="bg-gradient-to-br from-blue-500 to-blue-600 text-white p-6 rounde
          <h3 className="text-sm font-medium opacity-90">Total Portfolio Value</h3>
          <p className="text-3xl font-bold mt-2">
            ₹{(portfolioData.totalValue || 0).toLocaleString()}
          </p>
          <p className="text-sm opacity-75 mt-1">
            {portfolioData.receiptsCount || 0} active receipts
          </p>
        </div>

        <div className="bg-gradient-to-br from-green-500 to-green-600 text-white p-6 roun
          <h3 className="text-sm font-medium opacity-90">Available Credit</h3>
          <p className="text-3xl font-bold mt-2">
            ₹{(portfolioData.availableCredit || 0).toLocaleString()}
          </p>
          <p className="text-sm opacity-75 mt-1">80% of portfolio value</p>
        </div>

        <div className="bg-gradient-to-br from-purple-500 to-purple-600 text-white p-6 ro
          <h3 className="text-sm font-medium opacity-90">Ready for Loans</h3>
          <p className="text-3xl font-bold mt-2">{eligibleReceipts?.data?.length || 0}</p
          <p className="text-sm opacity-75 mt-1">Eligible receipts</p>
        </div>

        <div className="bg-gradient-to-br from-orange-500 to-orange-600 text-white p-6 ro
          <h3 className="text-sm font-medium opacity-90">Active Loans</h3>
          <p className="text-3xl font-bold mt-2">{activeLoans?.data?.length || 0}</p>
          <p className="text-sm opacity-75 mt-1">Outstanding loans</p>
        </div>
      </div>

      {/* Quick Actions */}
      <div className="bg-white rounded-xl shadow-sm border p-6">
        <h2 className="text-xl font-semibold mb-6">Quick Actions</h2>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
          <Button
            onClick={() => router.push('/deposit')}
            className="h-20 bg-blue-600 hover:bg-blue-700 flex flex-col"
          >
            <Plus className="w-6 h-6 mb-2" />
            New Deposit
          </Button>

          <Button
            onClick={() => router.push('/loans')}
            disabled={!eligibleReceipts?.data?.length}
            className="h-20 bg-green-600 hover:bg-green-700 flex flex-col"
            variant={eligibleReceipts?.data?.length ? "default" : "secondary"}
          >
            <CreditCard className="w-6 h-6 mb-2" />
            Apply for Loan
          </Button>

          <Button
```

```
            onClick={() => router.push('/receipts')}
            className="h-20 bg-purple-600 hover:bg-purple-700 flex flex-col"
          >
            <FileText className="w-6 h-6 mb-2" />
            View Receipts
          </Button>
        </div>
      </div>

      {/* Portfolio Holdings */}
      <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
        <div className="bg-white rounded-xl shadow-sm border p-6">
          <h2 className="text-xl font-semibold mb-6">Warehouse Receipts</h2>
          {portfolioData.receipts?.length > 0 ? (
            <div className="space-y-4">
              {portfolioData.receipts.map(receipt => (
                <div key={receipt.id} className="border rounded-lg p-4">
                  <div className="flex justify-between items-start">
                    <div>
                      <h3 className="font-medium">{receipt.receiptNumber}</h3>
                      <p className="text-sm text-gray-600">
                        {receipt.quantity} {receipt.unit} • {receipt.qualityGrade}
                      </p>
                    </div>
                    <div className="text-right">
                      <p className="font-semibold">₹{receipt.valuation?.toLocaleString()}
                      <span className="text-xs px-2 py-1 bg-green-100 text-green-700 roun
                        {receipt.status}
                      </span>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          ) : (
            <div className="text-center py-8">
              <FileText className="w-12 h-12 text-gray-400 mx-auto mb-4" />
              <p className="text-gray-500 mb-4">No warehouse receipts yet</p>
              <Button onClick={() => router.push('/deposit')}>
                Create Your First Deposit
              </Button>
            </div>
          )}
        </div>

        <div className="bg-white rounded-xl shadow-sm border p-6">
          <h2 className="text-xl font-semibold mb-6">Loan Opportunities</h2>
          {eligibleReceipts?.data?.length > 0 ? (
            <div className="space-y-4">
              {eligibleReceipts.data.map(receipt => (
                <div key={receipt.id} className="border rounded-lg p-4">
                  <div className="flex justify-between items-center">
                    <div>
                      <h3 className="font-medium">{receipt.receiptNumber}</h3>
                      <p className="text-sm text-gray-600">
                        Max Loan: ₹{receipt.availableLoanAmount?.toLocaleString()}
```

```
                    </p>
                  </div>
                  <Button
                    size="sm"
                    onClick={() => router.push(`/loans/apply?receiptId=${receipt.id}`)}
                  >
                    Apply Now
                  </Button>
                </div>
              </div>
            ))}
          </div>
        ) : (
          <div className="text-center py-8">
            <CreditCard className="w-12 h-12 text-gray-400 mx-auto mb-4" />
            <p className="text-gray-500">No eligible receipts for loans</p>
          </div>
        )}
      </div>
    </div>
  );
};
```

PHASE 3: LOAN MANAGEMENT SYSTEM - BANKING GRADE

3.1 Loan Application Flow

```
const LoanApplicationFlow = () => {
  const { receiptId } = useParams();
  const [selectedReceipt, setSelectedReceipt] = useState(null);
  const [loanAmount, setLoanAmount] = useState('');
  const [duration, setDuration] = useState('12');
  const [emi, setEmi] = useState(0);

  const { data: eligibleReceipts } = useQuery(['eligible-receipts'],
    () => fetch('/api/receipts/eligible-for-loans', { credentials: 'include' }).then(r =>
  );

  // Calculate EMI in real-time
  useEffect(() => {
    if (loanAmount && duration) {
      const principal = parseFloat(loanAmount);
      const months = parseInt(duration);
      const monthlyRate = 0.12 / 12; // 12% annual
      const calculatedEmi = (principal * monthlyRate * Math.pow(1 + monthlyRate, months))
                            (Math.pow(1 + monthlyRate, months) - 1);
      setEmi(calculatedEmi);
    }
  }, [loanAmount, duration]);
```

```
  const handleLoanApplication = async (e) => {
    e.preventDefault();

    try {
      const response = await fetch('/api/loans', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({
          receiptId: selectedReceipt.id,
          amount: parseFloat(loanAmount),
          durationMonths: parseInt(duration)
        })
      });

      const result = await response.json();

      if (result.success) {
        toast.success(`✅ Loan approved! Amount: ₹${loanAmount}`);
        queryClient.invalidateQueries(['active-loans']);
        queryClient.invalidateQueries(['portfolio']);
        router.push('/loans');
      } else {
        toast.error(result.error);
      }
    } catch (error) {
      toast.error('Application failed. Please try again.');
    }
  };

  return (
    <div className="max-w-4xl mx-auto p-6">
      <div className="bg-white rounded-xl shadow-sm border p-8">
        <h1 className="text-3xl font-bold mb-8">Apply for Loan</h1>

        <form onSubmit={handleLoanApplication} className="space-y-8">
          {/* Receipt Selection */}
          <div>
            <h2 className="text-xl font-semibold mb-4">Select Collateral Receipt</h2>
            <div className="grid gap-4">
              {eligibleReceipts?.data?.map(receipt => (
                <div
                  key={receipt.id}
                  className={`border rounded-lg p-4 cursor-pointer transition-all ${
                    selectedReceipt?.id === receipt.id
                      ? 'border-blue-500 bg-blue-50'
                      : 'border-gray-200 hover:border-gray-300'
                  }`}
                  onClick={() => setSelectedReceipt(receipt)}
                >
                  <div className="flex justify-between items-center">
                    <div>
                      <h3 className="font-medium">{receipt.receiptNumber}</h3>
                      <p className="text-sm text-gray-600">
                        Value: ₹{receipt.valuation?.toLocaleString()} •
                        Available: ₹{receipt.availableLoanAmount?.toLocaleString()}
```

```
                  </p>
                </div>
                {selectedReceipt?.id === receipt.id && (
                  <CheckCircle className="w-6 h-6 text-blue-500" />
                )}
              </div>
            </div>
          ))}
        </div>
      </div>

      {selectedReceipt && (
        <>
          {/* Loan Amount */}
          <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
            <div>
              <label className="block text-sm font-medium mb-2">Loan Amount *</label>
              <Input
                type="number"
                placeholder="Enter amount"
                value={loanAmount}
                onChange={(e) => setLoanAmount(e.target.value)}
                max={selectedReceipt.availableLoanAmount}
                required
              />
              <p className="text-sm text-gray-500 mt-1">
                Max: ₹{selectedReceipt.availableLoanAmount?.toLocaleString()}
              </p>
            </div>

            <div>
              <label className="block text-sm font-medium mb-2">Duration</label>
              <Select value={duration} onValueChange={setDuration}>
                <SelectTrigger>
                  <SelectValue />
                </SelectTrigger>
                <SelectContent>
                  <SelectItem value="6">6 months</SelectItem>
                  <SelectItem value="12">12 months</SelectItem>
                  <SelectItem value="18">18 months</SelectItem>
                  <SelectItem value="24">24 months</SelectItem>
                </SelectContent>
              </Select>
            </div>
          </div>

          {/* Loan Details */}
          {loanAmount && (
            <div className="bg-gray-50 rounded-lg p-6">
              <h3 className="font-semibold mb-4">Loan Details</h3>
              <div className="grid grid-cols-2 gap-4 text-sm">
                <div className="flex justify-between">
                  <span>Loan Amount:</span>
                  <span className="font-medium">₹{parseFloat(loanAmount).toLocaleStri
                </div>
                <div className="flex justify-between">
```

```
                    <span>Interest Rate:</span>
                    <span className="font-medium">12% per annum</span>
                  </div>
                  <div className="flex justify-between">
                    <span>Duration:</span>
                    <span className="font-medium">{duration} months</span>
                  </div>
                  <div className="flex justify-between">
                    <span>Monthly EMI:</span>
                    <span className="font-medium">₹{emi.toLocaleString()}</span>
                  </div>
                  <div className="flex justify-between border-t pt-2">
                    <span>Total Payable:</span>
                    <span className="font-medium">₹{(emi * parseInt(duration)).toLocale
                  </div>
                  <div className="flex justify-between">
                    <span>Total Interest:</span>
                    <span className="font-medium">₹{(emi * parseInt(duration) - parseFl
                  </div>
                </div>
              </div>
            )}

            {/* Submit */}
            <Button
              type="submit"
              size="lg"
              className="w-full"
              disabled={!loanAmount || parseFloat(loanAmount) > selectedReceipt.availab
            >
              Apply for Loan
            </Button>
          </>
        )}
      </form>
    </div>
  </div>
  );
};
```

3.2 Active Loans Management

```
const ActiveLoansManagement = () => {
  const { data: loansData } = useQuery(['active-loans'],
    () => fetch('/api/loans', { credentials: 'include' }).then(r => r.json())
  );

  const loans = loansData?.data || [];

  return (
    <div className="max-w-6xl mx-auto p-6">
      <h1 className="text-3xl font-bold mb-8">Loan Management</h1>

      {loans.length > 0 ? (
        <div className="space-y-6">
```

```
          {loans.map(loan => (
            <div key={loan.id} className="bg-white rounded-xl border shadow-sm p-6">
              <div className="flex justify-between items-start mb-6">
                <div>
                  <h3 className="text-xl font-semibold">Loan #{loan.id}</h3>
                  <p className="text-gray-600">
                    Against Receipt: {loan.collateralReceiptNumber}
                  </p>
                </div>
                <span className={`px-3 py-1 rounded-full text-sm font-medium ${
                  loan.status === 'active'
                    ? 'bg-green-100 text-green-700'
                    : 'bg-gray-100 text-gray-700'
                }`}>
                  {loan.status}
                </span>
              </div>

              <div className="grid grid-cols-2 md:grid-cols-4 gap-6 mb-6">
                <div>
                  <p className="text-sm text-gray-500">Loan Amount</p>
                  <p className="text-lg font-semibold">₹{loan.amount?.toLocaleString()}</
                </div>
                <div>
                  <p className="text-sm text-gray-500">Outstanding</p>
                  <p className="text-lg font-semibold text-red-600">
                    ₹{loan.outstandingAmount?.toLocaleString()}
                  </p>
                </div>
                <div>
                  <p className="text-sm text-gray-500">Monthly EMI</p>
                  <p className="text-lg font-semibold">₹{loan.monthlyEmi?.toLocaleString(
                </div>
                <div>
                  <p className="text-sm text-gray-500">Due Date</p>
                  <p className="text-lg font-semibold">
                    {new Date(loan.dueDate).toLocaleDateString()}
                  </p>
                </div>
              </div>

              {/* Progress Bar */}
              <div className="mb-6">
                <div className="flex justify-between text-sm text-gray-600 mb-2">
                  <span>Repayment Progress</span>
                  <span>
                    {Math.round(((loan.amount - loan.outstandingAmount) / loan.amount) *
                  </span>
                </div>
                <div className="w-full bg-gray-200 rounded-full h-2">
                  <div
                    className="bg-green-500 h-2 rounded-full transition-all duration-300"
                    style={{
                      width: `${((loan.amount - loan.outstandingAmount) / loan.amount) *
                    }}
                  />
```

```
              </div>
            </div>

            {/* Action Buttons */}
            <div className="flex gap-4">
              <Button className="bg-blue-600 hover:bg-blue-700">
                Make Payment
              </Button>
              <Button variant="outline">
                View Statement
              </Button>
              <Button variant="outline">
                Prepay Loan
              </Button>
            </div>
          </div>
        ))}
      </div>
    ) : (
      <div className="text-center py-12">
        <CreditCard className="w-16 h-16 text-gray-400 mx-auto mb-4" />
        <h2 className="text-xl font-semibold text-gray-900 mb-2">No Active Loans</h2>
        <p className="text-gray-600 mb-6">You don't have any active loans yet.</p>
        <Button onClick={() => router.push('/loans/apply')}>
          Apply for Your First Loan
        </Button>
      </div>
    )}
  </div>
);
};
```

PHASE 4: ENHANCED FEATURES

4.1 Cost Transparency Dashboard

```
const CostTransparencyDashboard = () => {
  const { data: costBreakdown } = useQuery(['cost-breakdown'],
    () => fetch('/api/costs/breakdown', { credentials: 'include' }).then(r => r.json())
  );

  const costs = costBreakdown?.data || {};

  return (
    <div className="max-w-4xl mx-auto p-6">
      <h1 className="text-3xl font-bold mb-8">Cost Breakdown</h1>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
        {/* Storage Costs */}
        <div className="bg-white rounded-xl border shadow-sm p-6">
          <h2 className="text-xl font-semibold mb-4">Storage Costs</h2>
          <div className="space-y-3">
            <div className="flex justify-between">
```

```
              <span className="text-gray-600">Monthly Storage Fee:</span>
              <span className="font-medium">₹{costs.monthlyStorage?.toLocaleString()}</sp
            </div>
            <div className="flex justify-between">
              <span className="text-gray-600">Quality Testing:</span>
              <span className="font-medium">₹{costs.qualityTesting?.toLocaleString()}</sp
            </div>
            <div className="flex justify-between">
              <span className="text-gray-600">Insurance:</span>
              <span className="font-medium">₹{costs.insurance?.toLocaleString()}</span>
            </div>
            <hr />
            <div className="flex justify-between font-semibold">
              <span>Total Storage Costs:</span>
              <span>₹{costs.totalStorage?.toLocaleString()}</span>
            </div>
          </div>
        </div>

        {/* Loan Costs */}
        <div className="bg-white rounded-xl border shadow-sm p-6">
          <h2 className="text-xl font-semibold mb-4">Loan Costs</h2>
          <div className="space-y-3">
            <div className="flex justify-between">
              <span className="text-gray-600">Interest Accrued:</span>
              <span className="font-medium">₹{costs.interestAccrued?.toLocaleString()}</s
            </div>
            <div className="flex justify-between">
              <span className="text-gray-600">Processing Fee:</span>
              <span className="font-medium">₹{costs.processingFee?.toLocaleString()}</spa
            </div>
            <div className="flex justify-between">
              <span className="text-gray-600">Late Payment Fee:</span>
              <span className="font-medium">₹{costs.lateFee?.toLocaleString()}</span>
            </div>
            <hr />
            <div className="flex justify-between font-semibold">
              <span>Total Loan Costs:</span>
              <span>₹{costs.totalLoanCosts?.toLocaleString()}</span>
            </div>
          </div>
        </div>
      </div>

      {/* Net Position */}
      <div className="bg-green-50 rounded-xl border border-green-200 p-6 mt-6">
        <h2 className="text-xl font-semibold text-green-900 mb-4">Your Net Position</h2>
        <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
          <div className="text-center">
            <p className="text-2xl font-bold text-green-700">
              ₹{costs.portfolioValue?.toLocaleString()}
            </p>
            <p className="text-green-600">Portfolio Value</p>
          </div>
          <div className="text-center">
            <p className="text-2xl font-bold text-red-600">
```

```
              -₹{costs.totalCosts?.toLocaleString()}
            </p>
            <p className="text-red-600">Total Costs</p>
          </div>
          <div className="text-center">
            <p className="text-2xl font-bold text-green-700">
              ₹{(costs.portfolioValue - costs.totalCosts)?.toLocaleString()}
            </p>
            <p className="text-green-600">Net Value</p>
          </div>
        </div>
      </div>
    </div>
  );
};
```

IMPLEMENTATION SUCCESS CRITERIA

✅ Core Functionality Working

- Deposit form creates commodity AND warehouse receipt immediately
- Dashboard shows real portfolio data and values
- Receipts appear in loans section as eligible collateral
- Loan application workflow completes successfully
- All database operations persist data correctly

✅ Banking-Grade UX

- Single-page streamlined deposit form with real-time calculations
- Portfolio dashboard like Zerodha with comprehensive data
- Professional loan management interface
- Cost transparency and billing breakdown
- Quick actions and intuitive navigation

✅ Business Logic Complete

- Seamless deposit → receipt → loan workflow
- Automatic loan eligibility calculation
- EMI calculation and payment tracking
- Collateral utilization management
- Portfolio performance monitoring

✅ Production Ready

- Working API endpoints with proper error handling
- Database schema optimized for performance
- Form validation and user feedback
- Mobile responsive design throughout
- Professional institutional-grade interface

IMPLEMENT THIS COMPLETE REBUILD TO CREATE A FULLY WORKING AGRICULTURAL FINTECH PLATFORM.