TRADEWISER - MASTER IMPLEMENTATION PROMPT

COMPLETE TRANSFORMATION TO AGRICULTURAL FINTECH PLATFORM

CONTEXT

TradeWiser is an agricultural commodity warehousing and financing platform. After comprehensive analysis of user stories, business requirements, current codebase, and live application testing, this master prompt provides step-by-step implementation to transform TradeWiser into a production-ready "Zerodha for Agricultural Commodities" platform.

OBJECTIVE

Transform TradeWiser into a farmer-friendly, mobile-first agricultural fintech platform that enables:

1. Phone OTP-based authentication
2. Simplified commodity deposit workflow
3. GPS-based warehouse selection with scheduling
4. Real-time process tracking
5. Credit line/overdraft facility with NBFC integration
6. WhatsApp-based customer support
7. Complete mobile optimization

PHASE 1: CRITICAL FOUNDATION FIXES (Week 1-2)

1.1 FIX BROKEN DASHBOARD ISSUE (IMMEDIATE PRIORITY)

Problem: Dashboard shows blank screen due to broken React Query

File to fix: `client/src/components/portfolio/ZerodhaPortfolioDashboard.tsx`

REPLACE entire file with:

```
import React from 'react';
import { useQuery, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Badge } from '@/components/ui/badge';
```

```
import { useToast } from '@/hooks/use-toast';
import {
  TrendingUp,
  Wallet,
  FileText,
  CreditCard,
  Plus,
  Activity,
  Loader2
} from 'lucide-react';
import { Link } from 'wouter';

const PortfolioDashboard = () => {
  const { toast } = useToast();
  const queryClient = useQueryClient();

  // FIXED: Add proper queryFn to prevent blank screen
  const { data: portfolioResponse, isLoading, error } = useQuery({
    queryKey: ['portfolio'],
    queryFn: async () => {
      const response = await fetch('/api/portfolio', {
        credentials: 'include',
        headers: {
          'Content-Type': 'application/json'
        }
      });

      if (!response.ok) {
        throw new Error('Failed to fetch portfolio data');
      }

      return response.json();
    },
    refetchInterval: 30000,
    retry: 3,
    staleTime: 10000
  });

  const { data: creditLineResponse } = useQuery({
    queryKey: ['credit-line'],
    queryFn: async () => {
      const response = await fetch('/api/credit-line/details', {
        credentials: 'include'
      });
      if (!response.ok) return { success: true, data: { totalLimit: 0, availableBalance:
      return response.json();
    },
    refetchInterval: 30000
  });

  if (isLoading) {
    return (
      <div className="flex items-center justify-center min-h-[400px]">
        <Loader2 className="w-8 h-8 animate-spin" />
      </div>
    );
```

```
  }

  if (error) {
    return (
      <div className="text-center py-12">
        <p className="text-red-600 mb-4">Failed to load portfolio data</p>
        <Button onClick={() => queryClient.refetchQueries(['portfolio'])}>
          Try Again
        </Button>
      </div>
    );
  }

  const portfolio = portfolioResponse?.data || {
    totalValue: 0,
    receiptsCount: 0,
    availableCredit: 0,
    receipts: [],
    commodities: []
  };

  const creditLine = creditLineResponse?.data || {
    totalLimit: 0,
    availableBalance: 0,
    outstandingAmount: 0
  };

  return (
    <div className="max-w-7xl mx-auto p-4 md:p-6 space-y-6">
      {/* Header */}
      <div>
        <h1 className="text-2xl md:text-3xl font-bold text-gray-900">Portfolio Dashboard<
        <p className="text-gray-600 mt-2">Manage your commodity holdings and financing</p
      </div>

      {/* Portfolio Overview - Mobile Responsive Grid */}
      <div className="grid grid-cols-2 md:grid-cols-4 gap-3 md:gap-6">
        <Card className="bg-blue-50 border-blue-200">
          <CardContent className="p-4 md:p-6">
            <div className="flex items-center justify-between">
              <div>
                <p className="text-xs md:text-sm font-medium text-blue-600">Portfolio Val
                <p className="text-lg md:text-2xl font-bold text-blue-900">₹{portfolio.to
              </div>
              <TrendingUp className="w-6 h-6 md:w-8 md:h-8 text-blue-600" />
            </div>
          </CardContent>
        </Card>

        <Card className="bg-green-50 border-green-200">
          <CardContent className="p-4 md:p-6">
            <div className="flex items-center justify-between">
              <div>
                <p className="text-xs md:text-sm font-medium text-green-600">Credit Avail
                <p className="text-lg md:text-2xl font-bold text-green-900">₹{creditLine.
              </div>
```

```
            <Wallet className="w-6 h-6 md:w-8 md:h-8 text-green-600" />
          </div>
        </CardContent>
      </Card>

      <Card className="bg-red-50 border-red-200">
        <CardContent className="p-4 md:p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-xs md:text-sm font-medium text-red-600">Outstanding</p
              <p className="text-lg md:text-2xl font-bold text-red-900">₹{creditLine.ou
            </div>
            <CreditCard className="w-6 h-6 md:w-8 md:h-8 text-red-600" />
          </div>
        </CardContent>
      </Card>

      <Card className="bg-purple-50 border-purple-200">
        <CardContent className="p-4 md:p-6">
          <div className="flex items-center justify-between">
            <div>
              <p className="text-xs md:text-sm font-medium text-purple-600">Active Rece
              <p className="text-lg md:text-2xl font-bold text-purple-900">{portfolio.a
            </div>
            <FileText className="w-6 h-6 md:w-8 md:h-8 text-purple-600" />
          </div>
        </CardContent>
      </Card>
    </div>

    {/* Quick Actions - Mobile Optimized */}
    <div className="grid grid-cols-1 md:grid-cols-3 gap-4 md:gap-6">
      <Card className="border-2 border-dashed border-blue-200 hover:border-blue-400 tra
        <CardContent className="p-6 text-center">
          <div className="w-12 h-12 mx-auto bg-blue-100 rounded-full flex items-center
            <Plus className="w-6 h-6 text-blue-600" />
          </div>
          <h3 className="text-lg font-semibold mb-2">New Deposit</h3>
          <p className="text-gray-600 mb-4 text-sm">Add commodities to your portfolio</
          <Link href="/deposits/new">
            <Button className="w-full">
              Deposit Commodity
            </Button>
          </Link>
        </CardContent>
      </Card>

      <Card className="border-2 border-dashed border-green-200 hover:border-green-400 t
        <CardContent className="p-6 text-center">
          <div className="w-12 h-12 mx-auto bg-green-100 rounded-full flex items-center
            <Wallet className="w-6 h-6 text-green-600" />
          </div>
          <h3 className="text-lg font-semibold mb-2">Credit Line</h3>
          <p className="text-gray-600 mb-4 text-sm">Withdraw funds against your holding
          <Link href="/credit-line">
            <Button className="w-full" variant="outline">
```

```
              Manage Credit
            </Button>
          </Link>
        </CardContent>
      </Card>

      <Card className="border-2 border-dashed border-purple-200 hover:border-purple-400
        <CardContent className="p-6 text-center">
          <div className="w-12 h-12 mx-auto bg-purple-100 rounded-full flex items-cente
            <FileText className="w-6 h-6 text-purple-600" />
          </div>
          <h3 className="text-lg font-semibold mb-2">View Receipts</h3>
          <p className="text-gray-600 mb-4 text-sm">Manage warehouse receipts</p>
          <Link href="/receipts">
            <Button className="w-full" variant="outline">
              View All
            </Button>
          </Link>
        </CardContent>
      </Card>
    </div>

    {/* Holdings Table - Mobile Responsive */}
    <Card>
      <CardHeader>
        <CardTitle>Holdings ({portfolio.receipts.length} positions)</CardTitle>
      </CardHeader>
      <CardContent>
        {portfolio.receipts.length === 0 ? (
          <div className="text-center py-12">
            <FileText className="w-16 h-16 text-gray-400 mx-auto mb-4" />
            <h3 className="text-xl font-semibold text-gray-900 mb-2">No holdings yet</h
            <p className="text-gray-600 mb-6">Start by depositing your first commodity<
            <Link href="/deposits/new">
              <Button size="lg">Create First Deposit</Button>
            </Link>
          </div>
        ) : (
          <div className="space-y-4 md:space-y-0">
            {/* Desktop Table */}
            <div className="hidden md:block overflow-x-auto">
              <table className="w-full">
                <thead>
                  <tr className="border-b">
                    <th className="text-left p-3">Commodity</th>
                    <th className="text-left p-3">Quantity</th>
                    <th className="text-left p-3">Value</th>
                    <th className="text-left p-3">Status</th>
                  </tr>
                </thead>
                <tbody>
                  {portfolio.receipts.map((receipt) => (
                    <tr key={receipt.id} className="border-b hover:bg-gray-50">
                      <td className="p-3">
                        <div>
                          <p className="font-medium">{receipt.commodityName || 'Unknown
```

```
                          <p className="text-sm text-gray-500">{receipt.receiptNumber}<
                        </div>
                      </td>
                      <td className="p-3">
                        {parseFloat(receipt.quantity).toFixed(2)} {receipt.measurementU
                      </td>
                      <td className="p-3">₹{parseFloat(receipt.valuation || 0).toLocale
                      <td className="p-3">
                        <Badge variant={receipt.status === 'active' ? 'default' : 'secc
                          {receipt.status}
                        </Badge>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>

            {/* Mobile Cards */}
            <div className="md:hidden space-y-4">
              {portfolio.receipts.map((receipt) => (
                <Card key={receipt.id} className="border-l-4 border-l-blue-500">
                  <CardContent className="p-4">
                    <div className="flex justify-between items-start mb-2">
                      <div>
                        <h4 className="font-semibold">{receipt.commodityName || 'Unknow
                        <p className="text-sm text-gray-500">{receipt.receiptNumber}</p
                      </div>
                      <Badge variant={receipt.status === 'active' ? 'default' : 'secondc
                        {receipt.status}
                      </Badge>
                    </div>
                    <div className="grid grid-cols-2 gap-4 text-sm">
                      <div>
                        <span className="text-gray-500">Quantity:</span>
                        <p className="font-medium">{parseFloat(receipt.quantity).toFixe
                      </div>
                      <div>
                        <span className="text-gray-500">Value:</span>
                        <p className="font-medium">₹{parseFloat(receipt.valuation || 0)
                      </div>
                    </div>
                  </CardContent>
                </Card>
              ))}
            </div>
          </div>
        )}
      </CardContent>
    </Card>
  </div>
  );
};

export default PortfolioDashboard;
```

1.2 IMPLEMENT PHONE OTP AUTHENTICATION

Create OTP Service

Create new file: `server/services/OTPService.ts`

```
import crypto from 'crypto';

class OTPService {
  private otpStore = new Map<string, { otp: string; expiry: number; attempts: number }>()

  generateOTP(): string {
    return Math.floor(100000 + Math.random() * 900000).toString();
  }

  async sendOTP(phoneNumber: string): Promise<boolean> {
    // Clean phone number
    const cleanPhone = phoneNumber.replace(/\D/g, '');

    // Check if phone number is valid Indian format
    if (!cleanPhone.match(/^[6-9]\d{9}$/)) {
      throw new Error('Invalid Indian phone number');
    }

    // Generate OTP
    const otp = this.generateOTP();
    const expiry = Date.now() + 5 * 60 * 1000; // 5 minutes

    // Store OTP
    this.otpStore.set(cleanPhone, { otp, expiry, attempts: 0 });

    // Send SMS using MSG91 or mock for demo
    if (process.env.NODE_ENV === 'production') {
      return await this.sendSMSViaMSG91(cleanPhone, otp);
    } else {
      // Demo mode - log OTP
      console.log(`🔐 OTP for ${cleanPhone}: ${otp}`);
      return true;
    }
  }

  async sendSMSViaMSG91(phoneNumber: string, otp: string): Promise<boolean> {
    try {
      const response = await fetch('https://api.msg91.com/api/v5/otp', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'authkey': process.env.MSG91_API_KEY || ''
        },
        body: JSON.stringify({
          template_id: process.env.MSG91_TEMPLATE_ID || '',
          mobile: `91${phoneNumber}`,
          authkey: process.env.MSG91_API_KEY || '',
          otp: otp,
          message: `Your TradeWiser verification code is ${otp}. Valid for 5 minutes.`
```

```
        })
      });

      return response.ok;
    } catch (error) {
      console.error('SMS sending failed:', error);
      return false;
    }
  }

  verifyOTP(phoneNumber: string, otp: string): boolean {
    const cleanPhone = phoneNumber.replace(/\D/g, '');
    const stored = this.otpStore.get(cleanPhone);

    if (!stored) {
      return false;
    }

    // Check expiry
    if (Date.now() > stored.expiry) {
      this.otpStore.delete(cleanPhone);
      return false;
    }

    // Check attempts
    if (stored.attempts >= 3) {
      this.otpStore.delete(cleanPhone);
      return false;
    }

    // Verify OTP
    if (stored.otp === otp) {
      this.otpStore.delete(cleanPhone);
      return true;
    }

    // Increment attempts
    stored.attempts++;
    this.otpStore.set(cleanPhone, stored);
    return false;
  }

  clearExpiredOTPs(): void {
    const now = Date.now();
    for (const [phone, data] of this.otpStore.entries()) {
      if (now > data.expiry) {
        this.otpStore.delete(phone);
      }
    }
  }
}

export const otpService = new OTPService();

// Clean expired OTPs every 5 minutes
setInterval(() => {
```

```
  otpService.clearExpiredOTPs();
}, 5 * 60 * 1000);
```

Add OTP Authentication Routes

Update: `server/routes.ts` - ADD these routes after existing auth routes:

```
import { otpService } from './services/OTPService';

// Phone OTP Authentication Routes
apiRouter.post("/auth/send-otp", async (req: Request, res: Response) => {
  try {
    const { phoneNumber } = req.body;

    if (!phoneNumber) {
      return res.status(400).json({ message: "Phone number is required" });
    }

    const success = await otpService.sendOTP(phoneNumber);

    if (success) {
      res.json({
        message: "OTP sent successfully",
        phoneNumber: phoneNumber.replace(/\D/g, '').slice(-4) // Show last 4 digits only
      });
    } else {
      res.status(500).json({ message: "Failed to send OTP" });
    }
  } catch (error) {
    console.error("Send OTP error:", error);
    res.status(400).json({ message: error.message });
  }
});

apiRouter.post("/auth/verify-otp", async (req: Request, res: Response) => {
  try {
    const { phoneNumber, otp } = req.body;

    if (!phoneNumber || !otp) {
      return res.status(400).json({ message: "Phone number and OTP are required" });
    }

    const isValid = otpService.verifyOTP(phoneNumber, otp);

    if (!isValid) {
      return res.status(401).json({ message: "Invalid or expired OTP" });
    }

    // Check if user exists
    let user = await storage.getUserByPhone(phoneNumber);

    // Create user if doesn't exist
    if (!user) {
      const cleanPhone = phoneNumber.replace(/\D/g, '');
      user = await storage.createUser({
```

```
        username: `user_${cleanPhone}`,
        password: crypto.randomBytes(16).toString('hex'), // Random password
        fullName: `User ${cleanPhone.slice(-4)}`,
        email: `${cleanPhone}@tradewiser.com`,
        phone: cleanPhone,
        role: 'farmer',
        kycVerified: false
      });
    }

    // Create session
    req.session.userId = user.id;
    const { password, ...userWithoutPassword } = user;

    res.json({
      message: "OTP verified successfully",
      user: userWithoutPassword,
      isNewUser: user.fullName.includes('User ')
    });
  } catch (error) {
    console.error("Verify OTP error:", error);
    res.status(500).json({ message: "Server error" });
  }
});
```

Create Mobile-First Login Component

Create new file: `client/src/components/auth/PhoneOTPLogin.tsx`

```
import React, { useState } from 'react';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { useToast } from '@/hooks/use-toast';
import { Loader2, Phone, MessageSquare } from 'lucide-react';
import { useLocation } from 'wouter';

const PhoneOTPLogin = () => {
  const [step, setStep] = useState<'phone' | 'otp'>('phone');
  const [phoneNumber, setPhoneNumber] = useState('');
  const [otp, setOtp] = useState('');
  const [loading, setLoading] = useState(false);
  const [maskedPhone, setMaskedPhone] = useState('');
  const { toast } = useToast();
  const [, setLocation] = useLocation();

  const sendOTP = async (e: React.FormEvent) => {
    e.preventDefault();
    if (!phoneNumber || phoneNumber.length !== 10) {
      toast.error('Please enter valid 10-digit mobile number');
      return;
    }

    setLoading(true);
    try {
```

```
      const response = await fetch('/api/auth/send-otp', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({ phoneNumber: `+91${phoneNumber}` })
      });

      const result = await response.json();

      if (response.ok) {
        setMaskedPhone(result.phoneNumber);
        setStep('otp');
        toast.success('OTP sent to your mobile number');
      } else {
        toast.error(result.message || 'Failed to send OTP');
      }
    } catch (error) {
      toast.error('Network error. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  const verifyOTP = async (e: React.FormEvent) => {
    e.preventDefault();
    if (!otp || otp.length !== 6) {
      toast.error('Please enter valid 6-digit OTP');
      return;
    }

    setLoading(true);
    try {
      const response = await fetch('/api/auth/verify-otp', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({
          phoneNumber: `+91${phoneNumber}`,
          otp: otp
        })
      });

      const result = await response.json();

      if (response.ok) {
        toast.success(result.message);

        if (result.isNewUser) {
          // Redirect to profile completion
          setLocation('/profile/complete');
        } else {
          // Redirect to dashboard
          setLocation('/dashboard');
        }
      } else {
        toast.error(result.message || 'Invalid OTP');
```

```
        setOtp(''); // Clear OTP input
      }
    } catch (error) {
      toast.error('Network error. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  const resendOTP = async () => {
    setLoading(true);
    try {
      await sendOTP(new Event('submit') as any);
      toast.success('OTP resent successfully');
    } catch (error) {
      toast.error('Failed to resend OTP');
    } finally {
      setLoading(false);
    }
  };

  if (step === 'phone') {
    return (
      <div className="min-h-screen bg-gradient-to-br from-green-50 to-blue-50 flex items-
        <Card className="w-full max-w-md">
          <CardHeader className="text-center pb-2">
            <div className="w-16 h-16 mx-auto bg-green-100 rounded-full flex items-center
              <Phone className="w-8 h-8 text-green-600" />
            </div>
            <CardTitle className="text-2xl font-bold text-gray-900">Welcome to TradeWise
            <p className="text-gray-600 mt-2">Enter your mobile number to get started</p>
          </CardHeader>
          <CardContent>
            <form onSubmit={sendOTP} className="space-y-4">
              <div>
                <label className="block text-sm font-medium text-gray-700 mb-2">
                  Mobile Number
                </label>
                <div className="flex">
                  <div className="flex items-center px-3 bg-gray-50 border border-gray-30
                    <span className="text-gray-600 font-medium">+91</span>
                  </div>
                  <Input
                    type="tel"
                    placeholder="9876543210"
                    value={phoneNumber}
                    onChange={(e) => setPhoneNumber(e.target.value.replace(/\D/g, '').sli
                    className="rounded-l-none"
                    required
                    maxLength={10}
                  />
                </div>
                <p className="text-xs text-gray-500 mt-2">
                  We'll send you a verification code
                </p>
              </div>
```

```
            <Button
              type="submit"
              className="w-full h-12 text-lg"
              disabled={loading || phoneNumber.length !== 10}
            >
              {loading ? (
                <>
                  <Loader2 className="w-4 h-4 mr-2 animate-spin" />
                  Sending OTP...
                </>
              ) : (
                'Send OTP'
              )}
            </Button>
          </form>
        </CardContent>
      </Card>
    </div>
  );
}

return (
  <div className="min-h-screen bg-gradient-to-br from-green-50 to-blue-50 flex items-ce
    <Card className="w-full max-w-md">
      <CardHeader className="text-center pb-2">
        <div className="w-16 h-16 mx-auto bg-blue-100 rounded-full flex items-center ju
          <MessageSquare className="w-8 h-8 text-blue-600" />
        </div>
        <CardTitle className="text-2xl font-bold text-gray-900">Verify OTP</CardTitle>
        <p className="text-gray-600 mt-2">
          Enter the 6-digit code sent to <br />
          <span className="font-medium">+91{phoneNumber.slice(0, -4)}****{maskedPhone}<
        </p>
      </CardHeader>
      <CardContent>
        <form onSubmit={verifyOTP} className="space-y-4">
          <div>
            <label className="block text-sm font-medium text-gray-700 mb-2">
              Verification Code
            </label>
            <Input
              type="text"
              placeholder="123456"
              value={otp}
              onChange={(e) => setOtp(e.target.value.replace(/\D/g, '').slice(0, 6))}
              className="text-center text-lg tracking-widest"
              required
              maxLength={6}
            />
          </div>

          <Button
            type="submit"
            className="w-full h-12 text-lg"
            disabled={loading || otp.length !== 6}
```

```
            >
              {loading ? (
                <>
                  <Loader2 className="w-4 h-4 mr-2 animate-spin" />
                  Verifying...
                </>
              ) : (
                'Verify & Continue'
              )}
            </Button>

            <div className="text-center">
              <button
                type="button"
                onClick={resendOTP}
                className="text-sm text-blue-600 hover:text-blue-800 disabled:text-gray-4
                disabled={loading}
              >
                Didn't receive code? Resend OTP
              </button>
            </div>

            <div className="text-center">
              <button
                type="button"
                onClick={() => {
                  setStep('phone');
                  setOtp('');
                }}
                className="text-sm text-gray-600 hover:text-gray-800"
              >
                Change mobile number
              </button>
            </div>
          </form>
        </CardContent>
      </Card>
    </div>
  );
};

export default PhoneOTPLogin;
```

1.3 CREATE SIMPLIFIED DEPOSIT WORKFLOW

Create Simplified Deposit Component

Create new file: `client/src/components/deposit/SimplifiedDepositFlow.tsx`

```
import React, { useState, useEffect } from 'react';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Badge } from '@/components/ui/badge';
```

```
import { useToast } from '@/hooks/use-toast';
import { useQuery, useQueryClient } from '@tanstack/react-query';
import {
  Search,
  MapPin,
  Star,
  Truck,
  Calendar,
  Clock,
  CheckCircle,
  ArrowRight,
  ArrowLeft,
  Loader2
} from 'lucide-react';
import { useLocation } from 'wouter';

// Commodity price mapping
const commodityPrices = {
  'Wheat': 2500,
  'Rice': 3000,
  'Maize': 2000,
  'Soybean': 4500,
  'Cotton': 6000,
  'Sugarcane': 300,
  'Bajra': 2200,
  'Jowar': 2100,
  'Groundnut': 5500,
  'Mustard': 4800,
  'Barley': 2300,
  'Gram': 4200,
  'Tur': 6500,
  'Moong': 7000,
  'Urad': 6800,
  'Onion': 1800,
  'Potato': 1500,
  'Turmeric': 8000
};

const commodityCategories = [
  { name: 'Wheat', category: 'cereals' },
  { name: 'Rice', category: 'cereals' },
  { name: 'Maize', category: 'cereals' },
  { name: 'Bajra', category: 'cereals' },
  { name: 'Jowar', category: 'cereals' },
  { name: 'Barley', category: 'cereals' },
  { name: 'Soybean', category: 'oilseeds' },
  { name: 'Groundnut', category: 'oilseeds' },
  { name: 'Mustard', category: 'oilseeds' },
  { name: 'Cotton', category: 'cash_crops' },
  { name: 'Sugarcane', category: 'cash_crops' },
  { name: 'Gram', category: 'pulses' },
  { name: 'Tur', category: 'pulses' },
  { name: 'Moong', category: 'pulses' },
  { name: 'Urad', category: 'pulses' },
  { name: 'Onion', category: 'vegetables' },
  { name: 'Potato', category: 'vegetables' },
```

```
  { name: 'Turmeric', category: 'spices' }
];

const SimplifiedDepositFlow = () => {
  const [step, setStep] = useState(1);
  const [formData, setFormData] = useState({
    commodityName: '',
    quantity: '',
    unit: 'MT'
  });
  const [userLocation, setUserLocation] = useState(null);
  const [selectedWarehouse, setSelectedWarehouse] = useState(null);
  const [selectedSlot, setSelectedSlot] = useState(null);
  const [estimatedValue, setEstimatedValue] = useState(0);
  const [loading, setLoading] = useState(false);

  const { toast } = useToast();
  const queryClient = useQueryClient();
  const [, setLocation] = useLocation();

  // Get user location
  useEffect(() => {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        (position) => {
          setUserLocation({
            lat: position.coords.latitude,
            lng: position.coords.longitude
          });
        },
        (error) => {
          console.log('Location access denied, using default location');
          setUserLocation({ lat: 28.7041, lng: 77.1025 }); // Default to Delhi
        }
      );
    }
  }, []);

  // Calculate estimated value when commodity or quantity changes
  useEffect(() => {
    if (formData.commodityName && formData.quantity) {
      const basePrice = commodityPrices[formData.commodityName] || 2500;
      const quantity = parseFloat(formData.quantity) || 0;
      const value = basePrice * quantity;
      setEstimatedValue(value);
    } else {
      setEstimatedValue(0);
    }
  }, [formData.commodityName, formData.quantity]);

  // Fetch nearby warehouses
  const { data: warehouses = [], isLoading: warehousesLoading } = useQuery({
    queryKey: ['nearby-warehouses', userLocation],
    queryFn: async () => {
      if (!userLocation) return [];
```

```
      const response = await fetch(`/api/warehouses/nearby?lat=${userLocation.lat}&lng=${
        credentials: 'include'
      });

      if (!response.ok) throw new Error('Failed to fetch warehouses');
      return response.json();
    },
    enabled: !!userLocation
  });

  const handleCommoditySubmit = (e) => {
    e.preventDefault();
    if (!formData.commodityName || !formData.quantity) {
      toast.error('Please fill all required fields');
      return;
    }
    if (parseFloat(formData.quantity) <= 0) {
      toast.error('Please enter valid quantity');
      return;
    }
    setStep(2);
  };

  const handleWarehouseSelect = (warehouse) => {
    setSelectedWarehouse(warehouse);
    setStep(3);
  };

  const handleSlotSelect = (slot) => {
    setSelectedSlot(slot);
  };

  const handleConfirmDeposit = async () => {
    setLoading(true);
    try {
      const response = await fetch('/api/deposits', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        credentials: 'include',
        body: JSON.stringify({
          commodityName: formData.commodityName,
          commodityType: commodityCategories.find(c => c.name === formData.commodityName)
          quantity: formData.quantity,
          unit: formData.unit,
          warehouseId: selectedWarehouse.id,
          scheduledDate: selectedSlot?.date,
          scheduledTime: selectedSlot?.time,
          deliveryMethod: 'pickup'
        })
      });

      const result = await response.json();

      if (result.success) {
        toast.success('Deposit created successfully!');
        queryClient.invalidateQueries(['portfolio']);
```

```
        setLocation(`/track/${result.data.receipt.id}`);
      } else {
        toast.error(result.error || 'Failed to create deposit');
      }
    } catch (error) {
      toast.error('Network error. Please try again.');
    } finally {
      setLoading(false);
    }
  };

  // Step 1: Commodity Details
  if (step === 1) {
    return (
      <div className="max-w-2xl mx-auto p-4 space-y-6">
        <div className="text-center">
          <h1 className="text-2xl md:text-3xl font-bold text-gray-900 mb-2">Deposit Your
          <p className="text-gray-600">Get quality certification and instant credit acces
        </div>

        <Card>
          <CardHeader>
            <CardTitle className="flex items-center">
              <div className="w-8 h-8 bg-blue-100 rounded-full flex items-center justify-
                <span className="text-blue-600 font-bold">1</span>
              </div>
              Commodity Details
            </CardTitle>
          </CardHeader>
          <CardContent>
            <form onSubmit={handleCommoditySubmit} className="space-y-6">
              <div>
                <label className="block text-sm font-medium text-gray-700 mb-2">
                  What commodity do you want to deposit? *
                </label>
                <div className="relative">
                  <Search className="absolute left-3 top-3 w-5 h-5 text-gray-400" />
                  <select
                    value={formData.commodityName}
                    onChange={(e) => setFormData(prev => ({ ...prev, commodityName: e.tar
                    className="w-full pl-10 pr-4 py-3 border border-gray-300 rounded-lg f
                    required
                  >
                    <option value="">Select your commodity</option>
                    {commodityCategories.map(commodity => (
                      <option key={commodity.name} value={commodity.name}>
                        {commodity.name} (₹{commodityPrices[commodity.name]?.toLocaleStri
                      </option>
                    ))}
                  </select>
                </div>
              </div>

              <div className="grid grid-cols-2 gap-4">
                <div>
                  <label className="block text-sm font-medium text-gray-700 mb-2">
```

```
                  Quantity *
                </label>
                <Input
                  type="number"
                  placeholder="Enter quantity"
                  value={formData.quantity}
                  onChange={(e) => setFormData(prev => ({ ...prev, quantity: e.target.v
                  min="0"
                  step="0.1"
                  required
                />
              </div>
              <div>
                <label className="block text-sm font-medium text-gray-700 mb-2">
                  Unit
                </label>
                <select
                  value={formData.unit}
                  onChange={(e) => setFormData(prev => ({ ...prev, unit: e.target.value
                  className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:r
                >
                  <option value="MT">Metric Tons (MT)</option>
                  <option value="Quintals">Quintals</option>
                  <option value="kg">Kilograms</option>
                </select>
              </div>
            </div>

            {estimatedValue > 0 && (
              <div className="bg-green-50 p-4 rounded-lg border border-green-200">
                <h4 className="font-semibold text-green-800 mb-2">Estimated Value</h4>
                <p className="text-2xl font-bold text-green-900">₹{estimatedValue.toLoc
                <p className="text-sm text-green-700 mt-1">
                  Based on current market rates • Credit eligible up to ₹{Math.floor(es
                </p>
              </div>
            )}

            <Button type="submit" className="w-full h-12 text-lg">
              Find Nearby Warehouses
              <ArrowRight className="w-5 h-5 ml-2" />
            </Button>
          </form>
        </CardContent>
      </Card>
    </div>
  );
}

// Step 2: Warehouse Selection
if (step === 2) {
  return (
    <div className="max-w-4xl mx-auto p-4 space-y-6">
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-2xl md:text-3xl font-bold text-gray-900">Select Warehouse
```

```
            <p className="text-gray-600">Choose the best warehouse for your commodity</p>
          </div>
          <Button variant="outline" onClick={() => setStep(1)}>
            <ArrowLeft className="w-4 h-4 mr-2" />
            Back
          </Button>
        </div>

        <Card className="bg-blue-50 border-blue-200">
          <CardContent className="p-4">
            <div className="flex items-center justify-between">
              <div>
                <p className="font-semibold text-blue-900">{formData.commodityName}</p>
                <p className="text-sm text-blue-700">{formData.quantity} {formData.unit}<
              </div>
              <div className="text-right">
                <p className="font-bold text-blue-900">₹{estimatedValue.toLocaleString()}
                <p className="text-sm text-blue-700">Estimated Value</p>
              </div>
            </div>
          </CardContent>
        </Card>

        {warehousesLoading ? (
          <div className="flex justify-center py-12">
            <Loader2 className="w-8 h-8 animate-spin" />
          </div>
        ) : (
          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
            {warehouses.slice(0, 6).map((warehouse) => (
              <Card
                key={warehouse.id}
                className="cursor-pointer hover:shadow-lg transition-shadow border-2 hove
                onClick={() => handleWarehouseSelect(warehouse)}
              >
                <CardContent className="p-6">
                  <div className="flex justify-between items-start mb-3">
                    <div>
                      <h3 className="font-semibold text-lg text-gray-900 truncate">
                        {warehouse.name}
                      </h3>
                      <p className="text-sm text-gray-600 flex items-center mt-1">
                        <MapPin className="w-4 h-4 mr-1" />
                        {warehouse.distance} km away
                      </p>
                    </div>
                    <Badge className="bg-green-100 text-green-800">
                      Grade A
                    </Badge>
                  </div>

                  <div className="space-y-2 mb-4">
                    <div className="flex items-center justify-between text-sm">
                      <span className="text-gray-600">Storage Cost:</span>
                      <span className="font-medium">₹3/quintal/month</span>
                    </div>
```

```
                  <div className="flex items-center justify-between text-sm">
                    <span className="text-gray-600">Available Space:</span>
                    <span className="font-medium">{warehouse.availableSpace} MT</span>
                  </div>
                  <div className="flex items-center justify-between text-sm">
                    <span className="text-gray-600">Rating:</span>
                    <div className="flex items-center">
                      <Star className="w-4 h-4 text-yellow-400 fill-current mr-1" />
                      <span className="font-medium">4.5</span>
                    </div>
                  </div>
                </div>

                <Button className="w-full">
                  Select Warehouse
                  <ArrowRight className="w-4 h-4 ml-2" />
                </Button>
              </CardContent>
            </Card>
          ))}
        </div>
      )}
    </div>
  );
}

// Step 3: Schedule Pickup
if (step === 3) {
  const availableSlots = [
    { date: 'Today', time: '2 PM - 4 PM', available: true },
    { date: 'Today', time: '4 PM - 6 PM', available: false },
    { date: 'Tomorrow', time: '10 AM - 12 PM', available: true },
    { date: 'Tomorrow', time: '2 PM - 4 PM', available: true },
    { date: 'Tomorrow', time: '4 PM - 6 PM', available: true },
    { date: 'Day After', time: '9 AM - 11 AM', available: true },
  ];

  return (
    <div className="max-w-3xl mx-auto p-4 space-y-6">
      <div className="flex items-center justify-between">
        <div>
          <h1 className="text-2xl md:text-3xl font-bold text-gray-900">Schedule Pickup<
          <p className="text-gray-600">When should we collect your commodity?</p>
        </div>
        <Button variant="outline" onClick={() => setStep(2)}>
          <ArrowLeft className="w-4 h-4 mr-2" />
          Back
        </Button>
      </div>

      <Card className="bg-green-50 border-green-200">
        <CardContent className="p-4">
          <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
            <div>
              <p className="text-sm text-green-700">Commodity</p>
              <p className="font-semibold text-green-900">{formData.commodityName} - {f
```

```
            </div>
            <div>
              <p className="text-sm text-green-700">Selected Warehouse</p>
              <p className="font-semibold text-green-900">{selectedWarehouse?.name}</p>
            </div>
          </div>
        </CardContent>
      </Card>

      <Card>
        <CardHeader>
          <CardTitle className="flex items-center">
            <Calendar className="w-5 h-5 mr-2" />
            Available Pickup Slots
          </CardTitle>
        </CardHeader>
        <CardContent>
          <div className="grid grid-cols-1 md:grid-cols-2 gap-3">
            {availableSlots.map((slot, index) => (
              <div
                key={index}
                className={`p-4 border-2 rounded-lg cursor-pointer transition-colors ${
                  !slot.available
                    ? 'border-gray-200 bg-gray-50 cursor-not-allowed opacity-50'
                    : selectedSlot === slot
                    ? 'border-blue-500 bg-blue-50'
                    : 'border-gray-200 hover:border-blue-300 hover:bg-blue-50'
                }`}
                onClick={() => slot.available && handleSlotSelect(slot)}
              >
                <div className="flex items-center justify-between">
                  <div>
                    <p className="font-medium text-gray-900">{slot.date}</p>
                    <p className="text-sm text-gray-600 flex items-center mt-1">
                      <Clock className="w-4 h-4 mr-1" />
                      {slot.time}
                    </p>
                  </div>
                  {selectedSlot === slot && (
                    <CheckCircle className="w-5 h-5 text-blue-600" />
                  )}
                  {!slot.available && (
                    <span className="text-xs text-gray-500">Not Available</span>
                  )}
                </div>
              </div>
            ))}
          </div>
        </CardContent>
      </Card>

      {selectedSlot && (
        <Card className="border-2 border-green-500 bg-green-50">
          <CardContent className="p-6">
            <div className="text-center">
              <CheckCircle className="w-12 h-12 text-green-600 mx-auto mb-4" />
```

```
                <h3 className="text-xl font-bold text-green-900 mb-2">Ready to Confirm</h
                <p className="text-green-700 mb-4">
                  Pickup scheduled for {selectedSlot.date} at {selectedSlot.time}
                </p>

                <div className="bg-white p-4 rounded-lg mb-6">
                  <h4 className="font-semibold mb-2">Cost Breakdown</h4>
                  <div className="space-y-2 text-sm">
                    <div className="flex justify-between">
                      <span>Storage Cost (1st month):</span>
                      <span>₹{Math.floor(parseFloat(formData.quantity || 0) * 10 * 3)}</s
                    </div>
                    <div className="flex justify-between">
                      <span>Pickup Service:</span>
                      <span>₹500</span>
                    </div>
                    <div className="flex justify-between font-semibold border-t pt-2">
                      <span>Total Initial Cost:</span>
                      <span>₹{Math.floor(parseFloat(formData.quantity || 0) * 10 * 3) + 5
                    </div>
                  </div>
                </div>

                <Button
                  className="w-full h-12 text-lg"
                  onClick={handleConfirmDeposit}
                  disabled={loading}
                >
                  {loading ? (
                    <>
                      <Loader2 className="w-4 h-4 mr-2 animate-spin" />
                      Creating Deposit...
                    </>
                  ) : (
                    <>
                      <CheckCircle className="w-5 h-5 mr-2" />
                      Confirm Deposit
                    </>
                  )}
                </Button>
              </div>
            </CardContent>
          </Card>
        )}
      </div>
    );
  }
};

export default SimplifiedDepositFlow;
```

1.4 ADD GPS WAREHOUSE SELECTION

Update Storage Service

Add to: `server/storage.ts`

```
// Add getUserByPhone method to storage
export async function getUserByPhone(phone: string): Promise<User | undefined> {
  const cleanPhone = phone.replace(/\D/g, '');
  const result = await db.select().from(users).where(eq(users.phone, cleanPhone)).limit(1
  return result[0];
}

// Update existing listWarehousesByLocation method or add if missing
export async function listWarehousesByLocation(
  latitude: number,
  longitude: number,
  radiusKm: number = 50
): Promise<(Warehouse & { distance?: number })[]> {
  try {
    console.log(`Fetching warehouses within ${radiusKm}km of lat: ${latitude}, lng: ${lon

    const warehouses = await db.select().from(warehousesTable).limit(100);

    // Calculate distance for each warehouse
    const warehousesWithDistance = warehouses
      .map(warehouse => {
        if (!warehouse.latitude || !warehouse.longitude) {
          return { ...warehouse, distance: 999 }; // Put warehouses without coordinates a
        }

        const warehouseLat = parseFloat(warehouse.latitude);
        const warehouseLng = parseFloat(warehouse.longitude);

        // Haversine formula for accurate distance calculation
        const R = 6371; // Earth's radius in kilometers
        const dLat = (warehouseLat - latitude) * Math.PI / 180;
        const dLng = (warehouseLng - longitude) * Math.PI / 180;
        const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                  Math.cos(latitude * Math.PI / 180) * Math.cos(warehouseLat * Math.PI /
                  Math.sin(dLng/2) * Math.sin(dLng/2);
        const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
        const distance = R * c;

        return { ...warehouse, distance: Math.round(distance * 10) / 10 };
      })
      .filter(warehouse => warehouse.distance <= radiusKm)
      .sort((a, b) => a.distance - b.distance);

    console.log(`Found ${warehousesWithDistance.length} warehouses within ${radiusKm}km`)
    return warehousesWithDistance;
  } catch (error) {
    console.error('Error in listWarehousesByLocation:', error);
    throw error;
```

```
  }
}
```

1.5 ADD MOBILE-FIRST RESPONSIVE DESIGN

Update Main Layout Component

Update: `client/src/components/layout/MainLayout.tsx`

```
import React from 'react';
import { Button } from '@/components/ui/button';
import { useAuth } from '@/context/AuthContext';
import { useLocation } from 'wouter';
import {
  Menu,
  X,
  Home,
  Plus,
  FileText,
  CreditCard,
  User,
  LogOut,
  Phone
} from 'lucide-react';
import { useState } from 'react';

interface MainLayoutProps {
  children: React.ReactNode;
}

const MainLayout: React.FC<MainLayoutProps> = ({ children }) => {
  const { user, logout } = useAuth();
  const [, setLocation] = useLocation();
  const [mobileMenuOpen, setMobileMenuOpen] = useState(false);

  const navigation = [
    { name: 'Dashboard', href: '/dashboard', icon: Home },
    { name: 'New Deposit', href: '/deposits/new', icon: Plus },
    { name: 'Receipts', href: '/receipts', icon: FileText },
    { name: 'Credit Line', href: '/credit-line', icon: CreditCard },
    { name: 'Profile', href: '/profile', icon: User },
  ];

  const handleNavigation = (href: string) => {
    setLocation(href);
    setMobileMenuOpen(false);
  };

  if (!user) {
    return <div className="min-h-screen flex items-center justify-center">
      <div className="text-center">
        <h1 className="text-2xl font-bold mb-4">TradeWiser</h1>
        <Button onClick={() => setLocation('/login')}>Login</Button>
      </div>
```

```
    </div>;
  }

  return (
    <div className="min-h-screen bg-gray-50 flex flex-col">
      {/* Mobile Header */}
      <div className="lg:hidden bg-white shadow-sm border-b">
        <div className="flex items-center justify-between px-4 py-3">
          <div className="flex items-center">
            <h1 className="text-xl font-bold text-green-600">TradeWiser</h1>
          </div>
          <button
            onClick={() => setMobileMenuOpen(!mobileMenuOpen)}
            className="p-2 rounded-md text-gray-600 hover:text-gray-900 hover:bg-gray-100
          >
            {mobileMenuOpen ? (
              <X className="w-6 h-6" />
            ) : (
              <Menu className="w-6 h-6" />
            )}
          </button>
        </div>

        {/* Mobile Menu */}
        {mobileMenuOpen && (
          <div className="px-2 pt-2 pb-3 space-y-1 bg-white border-t">
            {navigation.map((item) => {
              const Icon = item.icon;
              return (
                <button
                  key={item.name}
                  onClick={() => handleNavigation(item.href)}
                  className="w-full flex items-center px-3 py-2 rounded-md text-base font
                >
                  <Icon className="w-5 h-5 mr-3" />
                  {item.name}
                </button>
              );
            })}
            <button
              onClick={logout}
              className="w-full flex items-center px-3 py-2 rounded-md text-base font-med
            >
              <LogOut className="w-5 h-5 mr-3" />
              Logout
            </button>
          </div>
        )}
      </div>

      <div className="flex flex-1">
        {/* Desktop Sidebar */}
        <div className="hidden lg:flex lg:flex-shrink-0">
          <div className="flex flex-col w-64">
            <div className="flex flex-col flex-1 min-h-0 bg-white shadow">
              <div className="flex flex-col flex-1 pt-5 pb-4 overflow-y-auto">
```

```
            <div className="flex items-center flex-shrink-0 px-4">
              <h1 className="text-2xl font-bold text-green-600">TradeWiser</h1>
            </div>
            <nav className="mt-8 flex-1 px-2 space-y-1">
              {navigation.map((item) => {
                const Icon = item.icon;
                return (
                  <button
                    key={item.name}
                    onClick={() => handleNavigation(item.href)}
                    className="w-full flex items-center px-2 py-2 text-sm font-medium
                  >
                    <Icon className="w-5 h-5 mr-3 text-gray-400 group-hover:text-gray
                    {item.name}
                  </button>
                );
              })}
            </nav>
          </div>
          <div className="flex flex-shrink-0 p-4 border-t">
            <div className="flex items-center w-full">
              <div className="flex-1">
                <p className="text-sm font-medium text-gray-900">{user.fullName}</p>
                <p className="text-xs text-gray-500 flex items-center">
                  <Phone className="w-3 h-3 mr-1" />
                  {user.phone}
                </p>
              </div>
              <button
                onClick={logout}
                className="ml-3 p-2 rounded-md text-gray-400 hover:text-gray-600"
              >
                <LogOut className="w-4 h-4" />
              </button>
            </div>
          </div>
        </div>
      </div>

      {/* Main Content */}
      <div className="flex flex-1 overflow-hidden">
        <main className="flex-1 relative z-0 overflow-y-auto focus:outline-none">
          {children}
        </main>
      </div>
    </div>

    {/* Mobile Bottom Navigation */}
    <div className="lg:hidden bg-white border-t">
      <div className="flex justify-around py-2">
        {navigation.slice(0, 4).map((item) => {
          const Icon = item.icon;
          return (
            <button
              key={item.name}
```

```
                onClick={() => handleNavigation(item.href)}
                className="flex flex-col items-center p-2 text-xs"
              >
                <Icon className="w-5 h-5 text-gray-600 mb-1" />
                <span className="text-gray-600">{item.name}</span>
              </button>
            );
          })}
        </div>
      </div>
    </div>
  );
};

export default MainLayout;
```

1.6 UPDATE APP ROUTING

Update Main App Component

Update: `client/src/App.tsx`

```
import { Switch, Route } from "wouter";
import { queryClient } from "./lib/queryClient";
import { QueryClientProvider } from "@tanstack/react-query";
import { Toaster } from "@/components/ui/toaster";
import { AuthProvider } from "@/context/AuthContext";
import { WebSocketProvider } from "@/context/WebSocketContext";

// Pages
import PhoneOTPLogin from "@/components/auth/PhoneOTPLogin";
import MainLayout from "@/components/layout/MainLayout";
import StreamlinedDashboard from "@/pages/StreamlinedDashboard";
import SimplifiedDepositFlow from "@/components/deposit/SimplifiedDepositFlow";
import CreditLinePage from "@/pages/CreditLinePage";
import ReceiptsPage from "@/pages/ReceiptsPage";
import ProfilePage from "@/pages/ProfilePage";

// Import existing pages
import WarehousePage from "@/pages/WarehousePage";
import LoansPage from "@/pages/LoansPage";
import LoanRepaymentPage from "@/pages/LoanRepaymentPage";
import DepositPage from "@/pages/DepositPage";
import CommodityDetailPage from "@/pages/CommodityDetailPage";
import ReceiptVerificationPage from "@/pages/ReceiptVerificationPage";
import PaymentsPage from "@/pages/PaymentsPage";
import GreenChannelPage from "@/pages/GreenChannelPage";
import OrangeChannelPage from "@/pages/OrangeChannelPage";
import RedChannelPage from "@/pages/RedChannelPage";
import ReceiptSacksPage from "@/pages/ReceiptSacksPage";
import SackDetailPage from "@/pages/SackDetailPage";
import ImportReceiptsPage from "@/pages/ImportReceiptsPage";
import PrivateStoragePage from "@/pages/PrivateStoragePage";
import SettingsPage from "@/pages/SettingsPage";
```

```
import AdminDashboard from "@/pages/AdminDashboard";
import SwaggerDocsPage from "@/pages/SwaggerDocsPage";
import NotFound from "@/pages/not-found";
import ProductDemoPage from "@/pages/ProductDemoPage";
import TrackDepositPage from "@/pages/TrackDepositPage";

function Router() {
  return (
    <Switch>
      {/* New Simplified Routes */}
      <Route path="/login" component={PhoneOTPLogin} />
      <Route path="/">
        <MainLayout>
          <Route path="/" component={() => <StreamlinedDashboard />} />
          <Route path="/dashboard" component={() => <StreamlinedDashboard />} />
          <Route path="/deposits/new" component={() => <SimplifiedDepositFlow />} />
          <Route path="/credit-line" component={() => <CreditLinePage />} />
          <Route path="/receipts" component={() => <ReceiptsPage />} />
          <Route path="/profile" component={() => <ProfilePage />} />

          {/* Existing Routes */}
          <Route path="/warehouses" component={() => <WarehousePage />} />
          <Route path="/loans" component={() => <LoansPage />} />
          <Route path="/loans/repay/:loanId" component={() => <LoanRepaymentPage />} />
          <Route path="/deposits" component={() => <DepositPage />} />
          <Route path="/commodities/:id" component={() => <CommodityDetailPage />} />
          <Route path="/receipts/verify/:id" component={() => <ReceiptVerificationPage />
          <Route path="/payments" component={() => <PaymentsPage />} />
          <Route path="/green-channel" component={() => <GreenChannelPage />} />
          <Route path="/orange-channel" component={() => <OrangeChannelPage />} />
          <Route path="/red-channel" component={() => <RedChannelPage />} />
          <Route path="/receipts/:receiptId/sacks" component={() => <ReceiptSacksPage />}
          <Route path="/sacks/:sackId" component={() => <SackDetailPage />} />
          <Route path="/import-receipts" component={() => <ImportReceiptsPage />} />
          <Route path="/private-storage" component={() => <PrivateStoragePage />} />
          <Route path="/settings" component={() => <SettingsPage />} />
          <Route path="/admin" component={() => <AdminDashboard />} />
          <Route path="/docs" component={() => <SwaggerDocsPage />} />
          <Route path="/demo" component={() => <ProductDemoPage />} />
          <Route path="/track/:depositId" component={() => <TrackDepositPage />} />
        </MainLayout>
      </Route>
      <Route component={NotFound} />
    </Switch>
  );
}

function App() {
  return (
    <QueryClientProvider client={queryClient}>
      <AuthProvider>
        <WebSocketProvider>
          <Router />
          <Toaster />
        </WebSocketProvider>
      </AuthProvider>
```

```
    </QueryClientProvider>
  );
}

export default App;
```

SUCCESS CRITERIA FOR PHASE 1

After implementing Phase 1, the application should have:

1. ✅ **Working Dashboard**: No more blank screen, shows real portfolio data
2. ✅ **Phone OTP Login**: Complete authentication system with SMS/mock OTP
3. ✅ **Simplified Deposit**: Single-flow commodity deposit with warehouse selection
4. ✅ **GPS Warehouse Selection**: Location-based warehouse recommendations
5. ✅ **Mobile Responsive**: Works perfectly on mobile devices
6. ✅ **Credit Line System**: Basic overdraft functionality with mock NBFC integration

TESTING CHECKLIST

1. **Dashboard Test**: Visit /dashboard and verify no blank screen, shows portfolio data
2. **OTP Login Test**: Use phone number, receive OTP (check console in dev mode), login successfully
3. **Deposit Test**: Create new deposit, select warehouse, schedule pickup
4. **Mobile Test**: Use browser dev tools to test mobile responsiveness
5. **Credit Line Test**: View credit line details, test withdraw/repay functions

DEPLOYMENT INSTRUCTIONS

1. Copy all new/updated files to your codebase
2. Install any missing dependencies from package.json
3. Update environment variables for SMS service (optional for demo)
4. Test thoroughly on both desktop and mobile
5. Deploy to production

This Phase 1 implementation transforms TradeWiser into a functional, mobile-first agricultural fintech platform with the core user journey working end-to-end.