Frontend Fixes - Make It Actually Work

Package.json - Complete Dependencies

json

```
{
  "name": "yieldmax-frontend",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "test": "jest",
    "test:e2e": "playwright test"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "next": "^14.0.0",
    "styled-components": "^6.1.0",
    "framer-motion": "^10.16.0",
    "recharts": "^2.9.0",
    "lucide-react": "^0.292.0",
    "ethers": "^5.7.2",
    "wagmi": "^1.4.0",
    "viem": "^1.19.0",
    "@rainbow-me/rainbowkit": "^1.3.0",
    "axios": "^1.6.0",
    "@tanstack/react-query": "^5.0.0"
  },
  "devDependencies": {
    "@types/react": "^18.2.0",
    "@types/node": "^20.0.0",
    "@types/styled-components": "^5.1.0",
    "typescript": "^5.0.0",
    "@playwright/test": "^1.40.0",
    "@chainsafe/dappeteer": "^5.0.0",
```

```
    "jest": "^29.0.0",
    "@testing-library/react": "^14.0.0",
    "@testing-library/jest-dom": "^6.0.0"
  }
}
```

Missing Hooks Implementation

typescript

```tsx
// hooks/useWeb3.ts
import { useAccount, useConnect, useDisconnect, useNetwork, useSwitchNetwork } from 'wagm
import { useCallback } from 'react';

export const useWeb3 = () => {
  const { address: account, isConnected } = useAccount();
  const { connect, connectors } = useConnect();
  const { disconnect } = useDisconnect();
  const { chain } = useNetwork();
  const { switchNetwork } = useSwitchNetwork();

  const connectWallet = useCallback(async () => {
    const connector = connectors[0]; // MetaMask
    if (connector) {
      await connect({ connector });
    }
  }, [connect, connectors]);

  const switchChain = useCallback(async (chainId: number) => {
    if (switchNetwork) {
      await switchNetwork(chainId);
    }
  }, [switchNetwork]);

  return {
    account,
    chainId: chain?.id,
    isConnected,
    connectWallet,
    disconnect,
    switchChain
```

```
  };
};
```

typescript

```tsx
// hooks/useRealTimeData.ts
import { useQuery } from '@tanstack/react-query';
import { useState, useEffect } from 'react';
import axios from 'axios';

export const useRealTimeData = () => {
  const [gasPrice, setGasPrice] = useState(0);

  const { data: yields } = useQuery({
    queryKey: ['yields'],
    queryFn: async () => {
      const response = await axios.get('/api/yields');
      return response.data;
    },
    refetchInterval: 30000 // 30 seconds
  });

  const { data: portfolio } = useQuery({
    queryKey: ['portfolio'],
    queryFn: async () => {
      const response = await axios.get('/api/portfolio');
      return response.data;
    }
  });

  const { data: transactions } = useQuery({
    queryKey: ['transactions'],
    queryFn: async () => {
      const response = await axios.get('/api/transactions');
      return response.data;
    }
  });
```

```
  // Mock WebSocket for gas prices
  useEffect(() => {
    const interval = setInterval(() => {
      setGasPrice(Math.random() * 100 + 20);
    }, 5000);

    return () => clearInterval(interval);
  }, []);

  return {
    yields: yields || [],
    portfolio: portfolio || { positions: [], totalValue: 0, change24h: 0 },
    transactions: transactions || [],
    gasPrice,
    isConnected: true
  };
};
```

Missing Styled Components

typescript

```ts
// components/StyledComponents.ts
import styled from 'styled-components';
import { designTokens } from '../design-system';

export const SubTitle = styled.p`
  font-size: ${designTokens.typography.fontSize.lg};
  color: ${designTokens.colors.neutral[400]};
  margin-top: ${designTokens.spacing[2]};
`;

export const HeaderActions = styled.div`
  display: flex;
  gap: ${designTokens.spacing[4]};
  align-items: center;
`;

export const Logo = styled.div`
  display: flex;
  align-items: center;
  gap: ${designTokens.spacing[3]};
  margin-bottom: ${designTokens.spacing[8]};
`;

export const LogoIcon = styled.div`
  color: ${designTokens.colors.primary[500]};
`;

export const LogoText = styled.span`
  font-size: ${designTokens.typography.fontSize['2xl']};
  font-weight: ${designTokens.typography.fontWeight.bold};
  color: ${designTokens.colors.neutral[50]};
`;
```

```
export const Navigation = styled.nav`
  display: flex;
  flex-direction: column;
  gap: ${designTokens.spacing[2]};
  margin-bottom: ${designTokens.spacing[8]};
`;

export const NavItem = styled.a<{ active?: boolean }>`
  display: flex;
  align-items: center;
  gap: ${designTokens.spacing[3]};
  padding: ${designTokens.spacing[3]} ${designTokens.spacing[4]};
  border-radius: ${designTokens.borderRadius.base};
  color: ${({ active }) => active ? designTokens.colors.primary[400] : designTokens.color
  background: ${({ active }) => active ? designTokens.colors.glass.bg : 'transparent'};
  cursor: pointer;
  transition: all ${designTokens.animation.duration.base};

  &:hover {
    background: ${designTokens.colors.glass.bg};
    color: ${designTokens.colors.neutral[100]};
  }
`;

// Add all other missing styled components...
```

Missing Utility Functions

typescript

```typescript
// utils/format.ts
export const formatNumber = (num: number): string => {
  if (num >= 1e9) return `${(num / 1e9).toFixed(2)}B`;
  if (num >= 1e6) return `${(num / 1e6).toFixed(2)}M`;
  if (num >= 1e3) return `${(num / 1e3).toFixed(2)}K`;
  return num.toFixed(2);
};

export const formatCurrency = (num: number): string => {
  return new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD',
    minimumFractionDigits: 0,
    maximumFractionDigits: 2
  }).format(num);
};

export const formatPercentage = (num: number): string => {
  return `${num.toFixed(2)}%`;
};
```

App Wrapper Setup

typescript

```tsx
// pages/_app.tsx or app/layout.tsx
import '@rainbow-me/rainbowkit/styles.css';
import { RainbowKitProvider, getDefaultWallets } from '@rainbow-me/rainbowkit';
import { configureChains, createConfig, WagmiConfig } from 'wagmi';
import { mainnet, arbitrum, polygon, optimism } from 'wagmi/chains';
import { publicProvider } from 'wagmi/providers/public';
import { QueryClient, QueryClientProvider } from '@tanstack/react-query';
import { GlobalStyles } from '../yieldmax-ui-system';

const { chains, publicClient } = configureChains(
  [mainnet, arbitrum, polygon, optimism],
  [publicProvider()]
);

const { connectors } = getDefaultWallets({
  appName: 'YieldMax',
  projectId: 'YOUR_WALLETCONNECT_PROJECT_ID',
  chains
});

const wagmiConfig = createConfig({
  autoConnect: true,
  connectors,
  publicClient
});

const queryClient = new QueryClient();

export default function App({ Component, pageProps }: any) {
  return (
    <>
      <GlobalStyles />
      <WagmiConfig config={wagmiConfig}>
```

```
        <RainbowKitProvider chains={chains}>
          <QueryClientProvider client={queryClient}>
            <Component {...pageProps} />
          </QueryClientProvider>
        </RainbowKitProvider>
      </WagmiConfig>
    </>
  );
}
```