Step 1: Define Project Structure

Create the basic structure for your project. For simplicity, let's use React Native for the frontend and Node.js for the backend.

bash

```
# Frontend setup
npx react-native init RealTimeStockApp
cd RealTimeStockApp

# Backend setup
mkdir backend
cd backend
npm init -y
npm install express axios websocket
```

Step 2: Design the User Interface

Create a simple user interface for your app using React Native.

App.js (Frontend)

```
import React, { useEffect, useState } from 'react';
import { SafeAreaView, StyleSheet, FlatList, Text } from 'react-native';
import axios from 'axios';

const App = () => {
  const [stocks, setStocks] = useState([]);

  useEffect(() => {
    const fetchData = async () => {
      const response = await axios.get('http://localhost:3000/stocks');
      setStocks(response.data);
    };
    fetchData();
  }, []);

  return (
    <SafeAreaView style={styles.container}>
      <FlatList
        data={stocks}
        keyExtractor={(item) => item.symbol}
        renderItem={({ item }) => (
          <Text style={styles.item}>
            {item.symbol}: {item.price}
          </Text>
        )}
      />
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  item: {
    fontSize: 18,
    margin: 10,
  },
});
```

```
export default App;
```

Step 3: Set Up the Backend

Create a simple Node.js server to provide real-time stock data.

index.js (Backend)

```
const express = require('express');
const axios = require('axios');
const WebSocket = require('ws');

const app = express();
const port = 3000;

let stockData = [];

// Replace with your stock API key and endpoint
const stockApiKey = 'YOUR_API_KEY';
const stockApiUrl = 'https://api.example.com/stocks';

// Fetch stock data from the API
const fetchStockData = async () => {
  try {
    const response = await axios.get(stockApiUrl, {
      headers: { 'Authorization': `Bearer ${stockApiKey}` }
    });
    stockData = response.data;
  } catch (error) {
    console.error('Error fetching stock data:', error);
  }
};

// Periodically fetch stock data
setInterval(fetchStockData, 10000);

app.get('/stocks', (req, res) => {
  res.json(stockData);
});

const server = app.listen(port, () => {
  console.log(`Server running on http://localhost:${port}`);
});

// WebSocket server for real-time updates
const wss = new WebSocket.Server({ server });

wss.on('connection', (ws) => {
  console.log('Client connected');
  ws.send(JSON.stringify(stockData));

  // Send updates every 10 seconds
  const interval = setInterval(() => {
    ws.send(JSON.stringify(stockData));
  }, 10000);

  ws.on('close', () => {
    clearInterval(interval);
    console.log('Client disconnected');
  });
});
```

Step 4: Connect Frontend with Backend

Connect the frontend with the backend to display real-time stock updates.

App.js (Frontend)

```
import React, { useEffect, useState } from 'react';
import { SafeAreaView, StyleSheet, FlatList, Text } from 'react-native';
import { WebSocket } from 'ws';

const App = () => {
  const [stocks, setStocks] = useState([]);

  useEffect(() => {
    const ws = new WebSocket('ws://localhost:3000');

    ws.onmessage = (event) => {
      setStocks(JSON.parse(event.data));
    };

    return () => ws.close();
  }, []);

  return (
    <SafeAreaView style={styles.container}>
      <FlatList
        data={stocks}
        keyExtractor={(item) => item.symbol}
        renderItem={({ item }) => (
          <Text style={styles.item}>
            {item.symbol}: {item.price}
          </Text>
        )}
      />
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  },
  item: {
    fontSize: 18,
    margin: 10,
  },
});
```

```
export default App;
```

Step 5: Test and Deploy

Test your app thoroughly and deploy the backend to a cloud service like AWS or Heroku. Publish your app on the App Store/Google Play Store.

This guide should get you started on creating a real-time stock app. Feel free to customize and expand upon it to fit your specific needs. Happy coding! If you need more details on any part, let me know!