React-Native Developer - Code Challenge

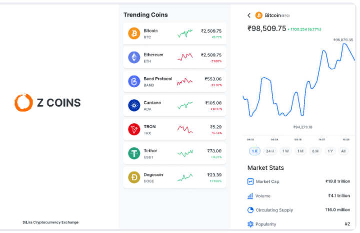
Objective:

Develop a Mobile application to connect to Binance (Or other) API, fetch all assets, and display them in a live-updating list.

Figma link:

BiLira Case Design

Created with Figma



Features:

- API Integration:
  - Connect to the Binance API(or any other market api is also acceptable, you can find all online) to fetch asset data.
- Live Data Updates:
  - Implement live updates for asset data.
  - Highlight price changes with colors:
    - Red for a decrease in value (**flashing for one second**).
    - Green for an increase in value (**flashing for one second**).
- Splash screen
- Home screen
- Asset detail screen (This screen is optional, but it's our pleasure to check out what you do here.)
  - Asset price
  - Asset chart with a timescale

  - Market Stats.

List Display:

  - Create a list to display the asset data. (See design)
  - List should include:
    - Icon, Asset name, symbol
    - 24h sparkline chart (if **positive**, green. if **negative** red. **default** color otherwise.) (It does not have to be live)
    - Price (**live data**, highlight on change)
    - 24h change (**live data**) (if positive, green. if negative red. Default color otherwise.)
  - Infinite scrolling (**render new assets as you scroll**)

Technical Requirements:

- Project Setup:
  - Follow recommended react-native setup flow.
  - Ensure the project is set up with **TypeScript**.
- Code Quality:
  - Adhere to **SOLID** principles.
  - Maintain consistent and meaningful **naming conventions**.
  - Include comprehensive code **documentation**.
- Testing:
  - Implement **unit tests** for critical components and functions.

Submission:

- Provide a public GitHub repository link with the complete source code.
- Publish on Expo for a small demo
- Ensure the repository includes a README with setup instructions and any other relevant information.

Evaluation Criteria:

- Correctness and completeness of the functionality.

- Focusing on the best practices, code quality, readability, and documentation (**JSDoc, Readme**).
- Efficient use of TypeScript.
- Proper implementation of live data updates with visual feedback.