Stocks Portfolio

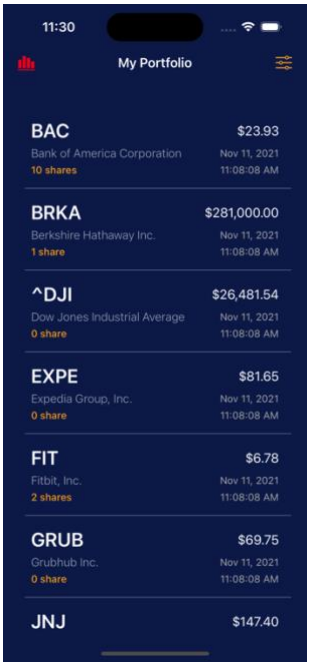
Introduction:

A quick documentation to explain the features in various screens and overall app.

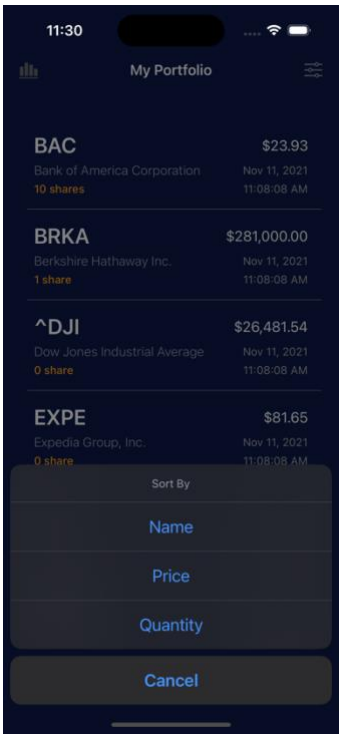
1. General App Documentation:
   Compatibility:
   - Tested on iPhone 8 phone device, iPhone 14 pro simulator
   - Supports portrait & landscape orientation.

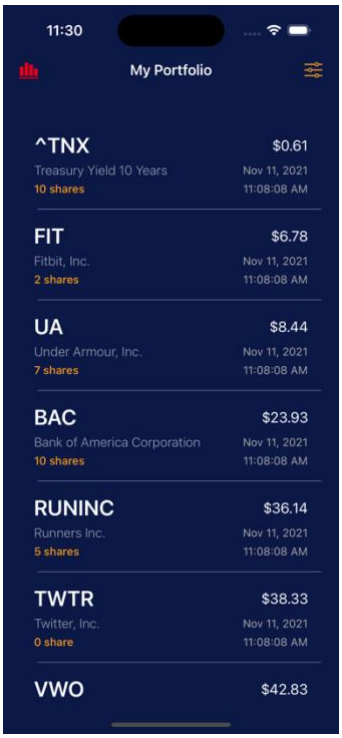
   Accessibility:
   - Ensured basic voice over is supported.

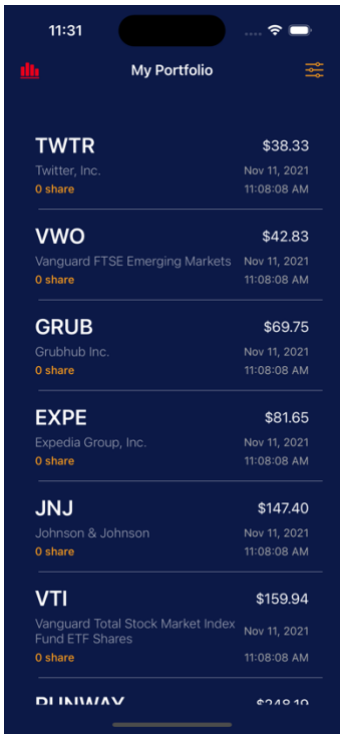
2. Programming Checks:
   - All UI was programmatically added for easier review.
   - Used SWIFT & SWIFTUI programming languages
   - Used Model View Controller design pattern.



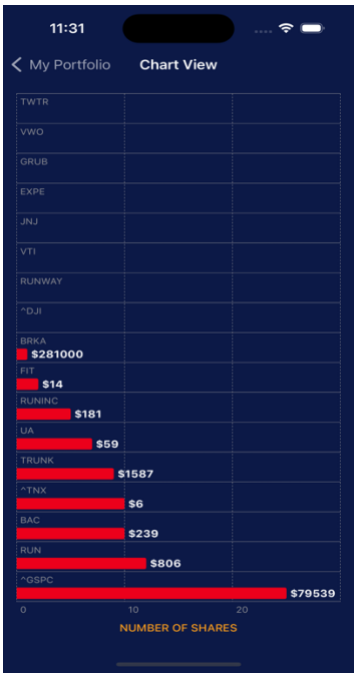
3. Main View:
   * Used custom app Icon.
   * Displays list of stocks with ticker name, name of the company invested, number of shares & price of each stock with current price and time.
   * The list is sorted by company name by default.
   * Navigation bar contains “Sort By” button on the right side and “Chart” button on the left side.



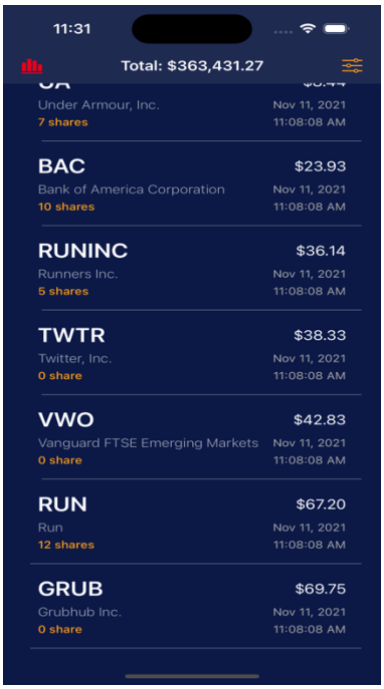




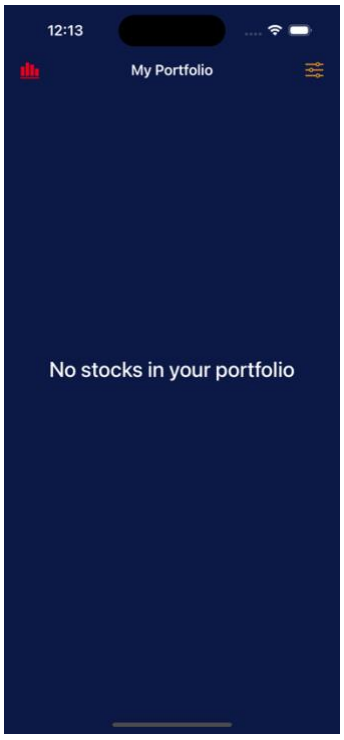
2. Sort By:
    - Upon clicking on the “Name” the list is sorted by company name.
    - Upon clicking on the “Price” the list is sorted by the stock price.
    - Upon clicking on the “Quantity” the list is sorted by number of shares.



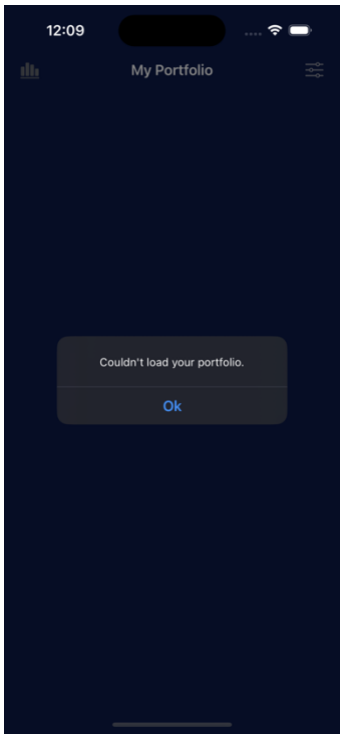
3. Chart View:
    - Chart View displays user’s stocks with number of shares on the X-Axis, Company ticker names on the Y-Axis.
    - Chart View also displays the total share value (number of shares * quantity) for each company.



4. Navigation Bar Title:
    - Total share value (number of shares * quantity for all the companies) is displayed upon scrolling.
    - “My portfolio” is displayed by default and if scrolled back to top.



5. For Empty Data URL:
    - UI updates to hide table view & display “No stocks in your portfolio” error text.



6. For Malformed URL:
    - UI updates to display an alert message with the text – “Couldn’t load your portfolio”.

7. How to run with empty data URL:
    - In NetworkManager.swift in fetchData method change the url to Constants.dataEmptyURL

```
func fetchData(completion: StockCompletionClosure?) {
    guard let url = Constants.dataEmptyUrl else { return }
    guard let request = createRequest(for: url) else {
        completion?(nil, NetworkError.invalidUrl)
           return
       }
       executeRequest(request: request, completion: completion)
}
```

8. How to run with malformed URL:
    - In NetworkManager.swift in fetchData method change the url to Constants.dataMalformedUrl

```
func fetchData(completion: StockCompletionClosure?) {
    guard let url = Constants.dataMalformedUrl else { return }
    guard let request = createRequest(for: url) else {
        completion?(nil, NetworkError.invalidUrl)
           return
       }
       executeRequest(request: request, completion: completion)
}
```

8. How to run with data URL:
   - In NetworkManager.swift in fetchData method change the url to Constants.dataUrl

```swift
func fetchData(completion: StockCompletionClosure?) {
    guard let url = Constants.dataUrl else { return }
    guard let request = createRequest(for: url) else {
        completion?(nil, NetworkError.invalidUrl)
           return
       }
       executeRequest(request: request, completion: completion)
}
```