ZCrypto

Introduction

ZCrypto is an app that allows users to view the live prices of various crypto currencies and add them to their custom portfolio. The app allows users to create an account on the app using Firebase Authentication API and Sign Into their custom portfolio.

Technologies Used

- Firebase Auth
- Firebase Database
- Combine Framework (Custom Publishers and Subscribers for asynchronous downloading of Crypto Currency Data)

Evolution (Start)

I personally an a really big crypto enthusiast and crypto trader, which motivated me to create an app around this topic. When I started working on this app it seemed fairly straightforward, the app required basic authentication functionality, after some research and reading firebase authentication documentation, though there were some other interesting options I went with firebase authentication to use in my app.

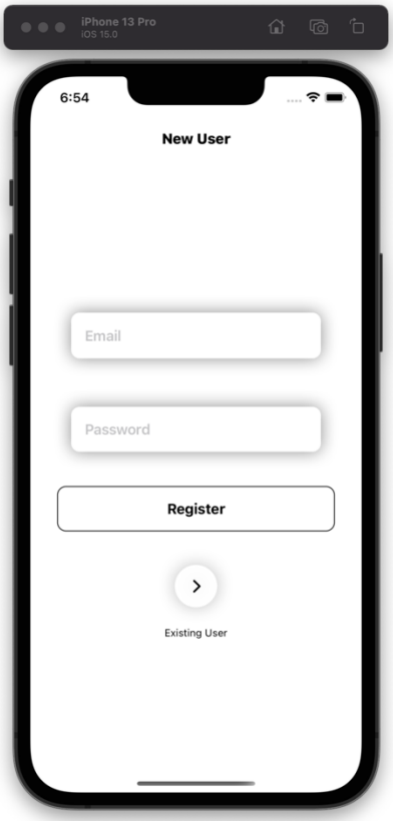
APP WORKING

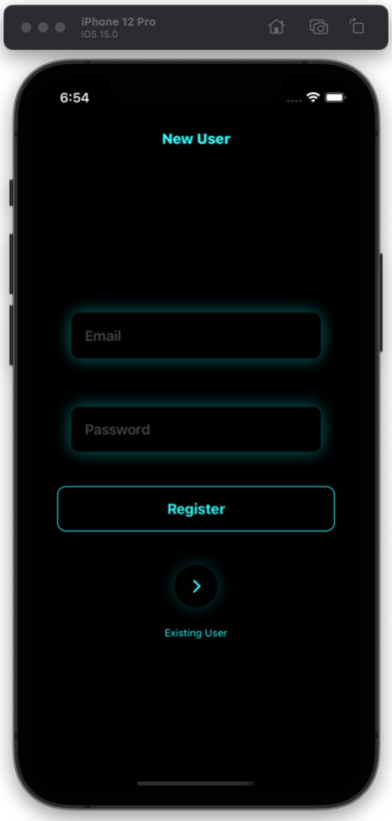
Start Screen:

When users launch the app, they are presented with a registration screen for new users and an option to sign in for existing users.

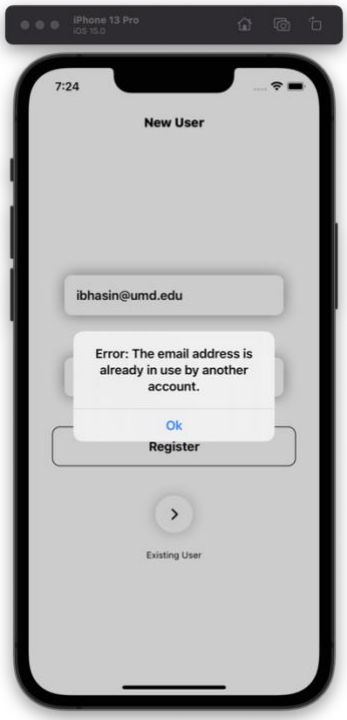
This makes the use of Firebase Authentication API to register new users based on unique email and authenticate existing users into the app if they enter correct combination of email and password registered in the database.

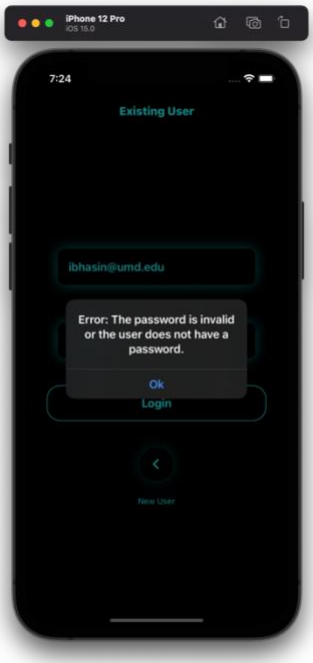
The app uses custom color theme for light and dark mode and is completely optimized in both the modes.





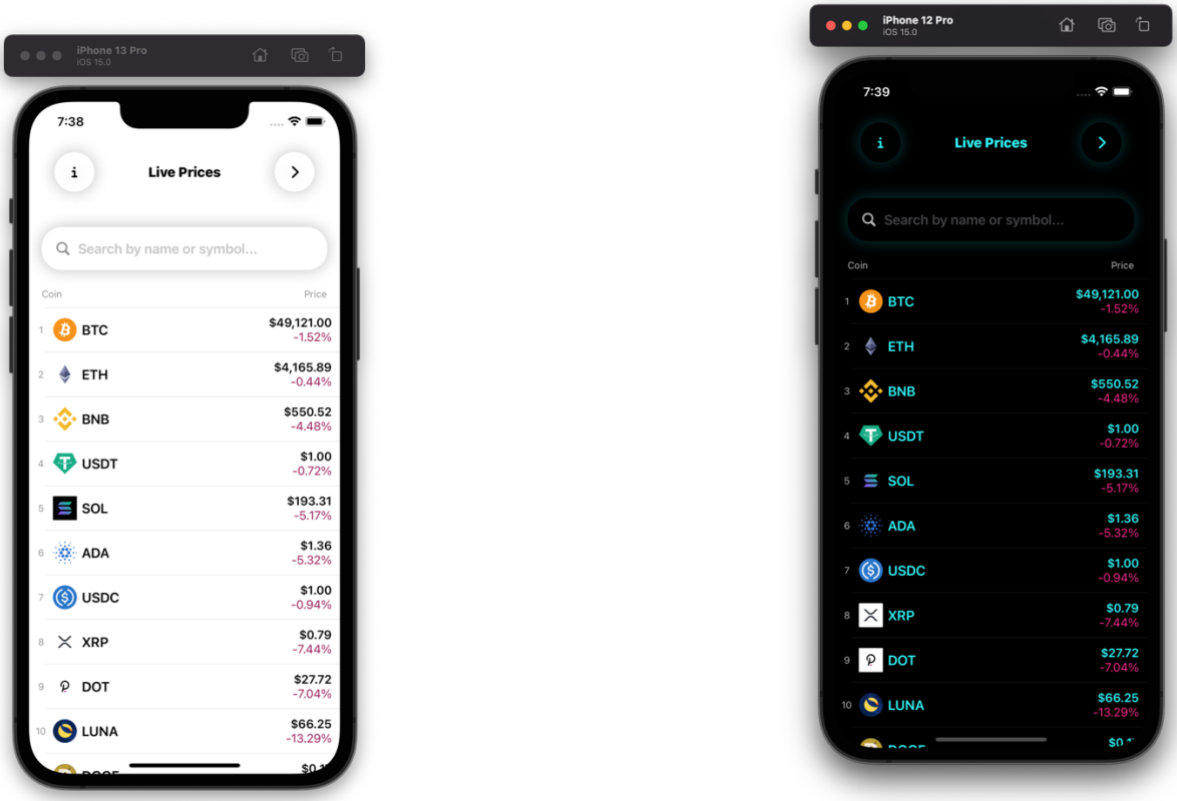
Errors Associated in authentication





The app displays custom error message when there is an error creating or signing in users.

Post Login Screen



When the user successfully login into the app they are presented with Live Prices view, which contain several elements.

Header: The header is a custom navigation bar that contains the Tittle of the page and navigation links that helps user to navigate through the app.

Search Bar: It's a custom search bar that helps user search for their favorite crypto currency based on its name, id or symbol. It fetches all the relevant matches and displays the results in the coins list.

Coins List: It's a custom list of crypto currency downloaded from an API (more details latter) and displays its live prices.

Evolution (Downloading Coin data and Coin Image)

This was probably the most difficult aspect of this app. The main goal was to find an API that provides all the necessary information for the app to use such as coin image or logo, live prices and percentage change in the value over a days or weeks' time.

After some research I came across as website called Coin Gecko https://api.coingecko.com, which had all the relevant data I required. The website provides coin data in JSON format upon successful API call. Then I began my research for an efficient way to download all the coin data into the app.

I found several ways to do this but the most efficient way I found to do this task was to use Apples latest Combine framework and custom publishers and subscribers. I created a Network Manager class to handle all asynchronous downloading of data using the Combine Framework. This class has a download method that takes in a URL as parameter and downloads the JSON data and handles any response errors.

The data downloaded was in JSON format, so I needed a coin model that had all the entry points returned from the JSON download and then I used a JSON decoder to convert the JSON data to coin model.

In view model I created a custom coin list publisher that subscribed to the data coming from the API and looked for any change in the prices of the crypto currency. So, whenever the prices change the publisher update the coins and this publisher updates the UI.

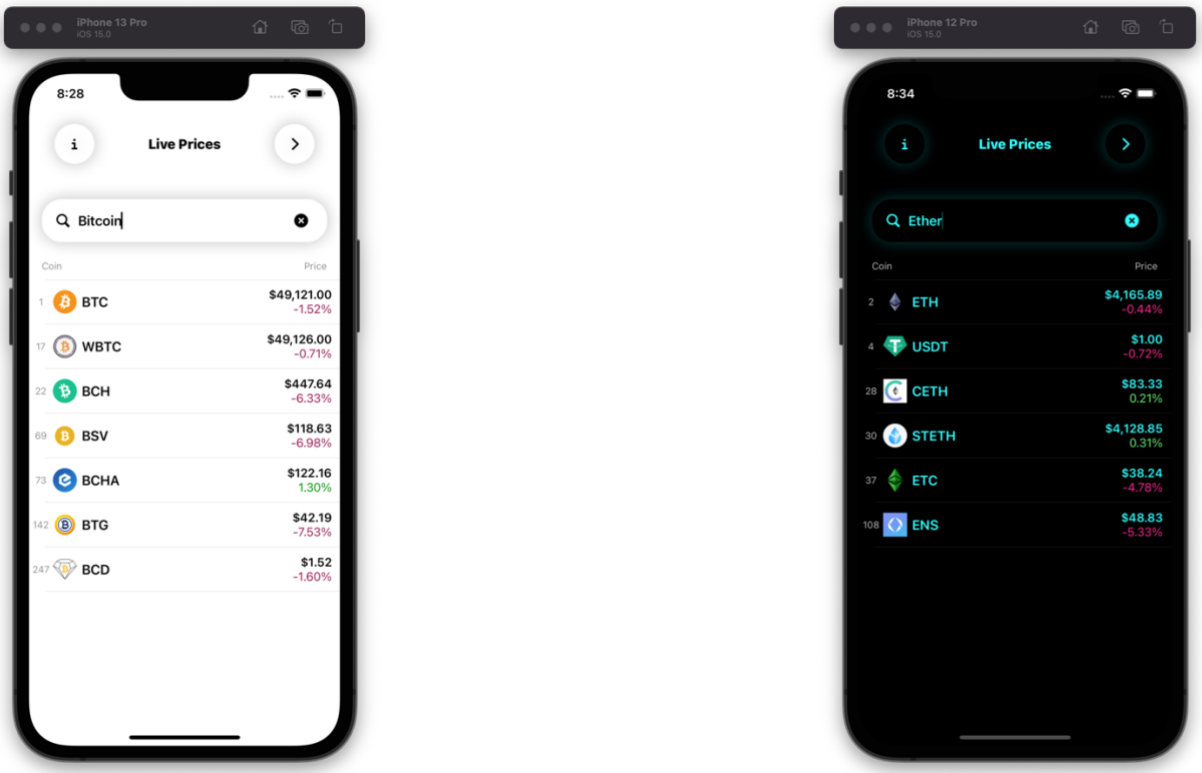
Another problem that I faced during the process was the downloading of Images. Rest of the data came from the API and was fairly small to handle and had a field called image that contained the URL for the image of the coins. I had to create separate publishers and subscribers to take the URL of image from each coin and download it separately. Since the prices of the coins change frequently but the image or logo barely change, this step was necessary for the app to work efficiently. I used the file manager to save the images, so they don't download every time the data is changed.

Both downloading of coin data and coin image takes advantage of network manager class and Combine framework to make this process really efficient.

Evolution (Search Bar)

This was one of the recent changes to the app. I found if a user needs to look for a particular coin, they have to scroll through the whole coins list to find their desired coin which made it have a bad user experience.

I went on to create a custom search bar, which filters out the result form the coin list and displays the matched results based on coin id, symbol or the coin name. I used a really neat trick to make this process efficient. When there is no data in the search bar it displays data for all coins and as soon as a string is entered into the search bar, I combined both coin list publisher and search bar publisher together to get the desired results.



The image on the left gets all the matched results for Bitcoin and the image on the right gets all the matched results for ether.

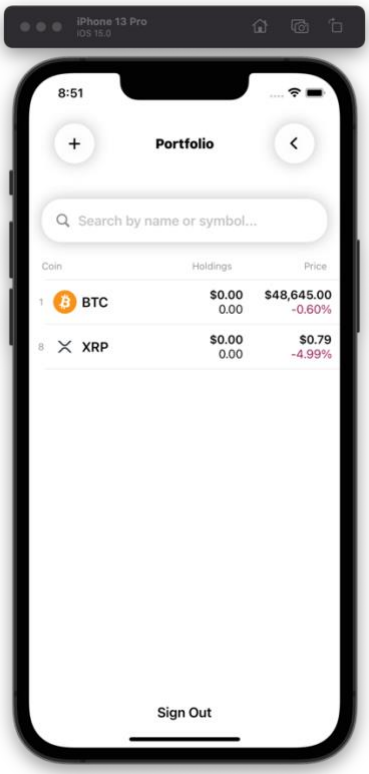
Portfolio Screen

The portfolio screen gives users an option to add coins to their portfolio. When users click on the add button, they are presented with a sheet view that contains a horizontal list of the coins and its logos. Users can either scroll through the coins or search them using the same search bar. Users can select their desired coins and press the save button to add the coins to their portfolio. Once the coin is added to their portfolio, they will appear in the portfolio view.

Evolution (Portfolio View)

This was one of the most difficult aspects of the app. When the add button is pressed a sheet is presented to add coins to the portfolio view, it took me some time to figure out that sheets have their own environment and I had to create new environment variables inside the sheet to get access to the coin data from the view model.

When the save button is pressed the coin data is uploaded to the firebase real time data base and upon user sign in the data is fetched from the data base to get portfolio coins.



At the bottom of the portfolio view users have an option to sign out and they are taken back to the registration/ login view.