How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: “**Capstone_Stage1**”
3. Replace the text in green

GitHub Username: kondeg

DividendPayout

Description

DividendPayout is an application intended to track cash flow from investments. The application will track year to date dividend payments, monthly dividend payments for both the past and the future.

Intended User

The application is intended as a budgeting tool for a retiree or an investor to determine the amount of dividend income that their portfolio generates.

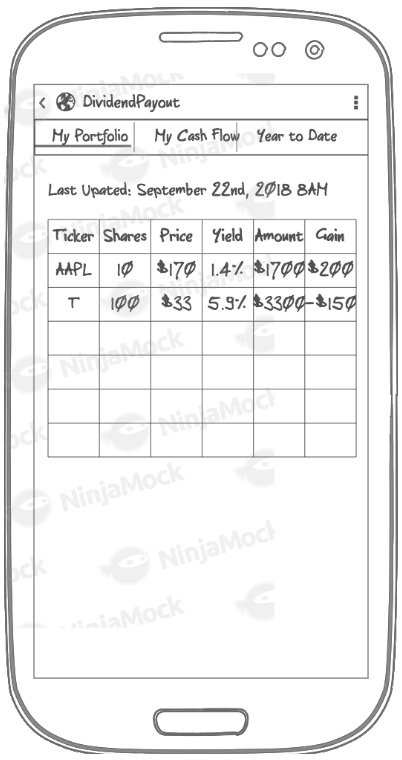
Features

- Application will be written using Java language.
- Saves list of stock tickers.
- Pulls dividend info for each ticker from an API
- Calculates and displays dividend payments by month
- Predicts cash flow for future months based on historical data
- App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
- App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.
- App uses a SyncAdapter to sync local stock data cache with stock API

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1

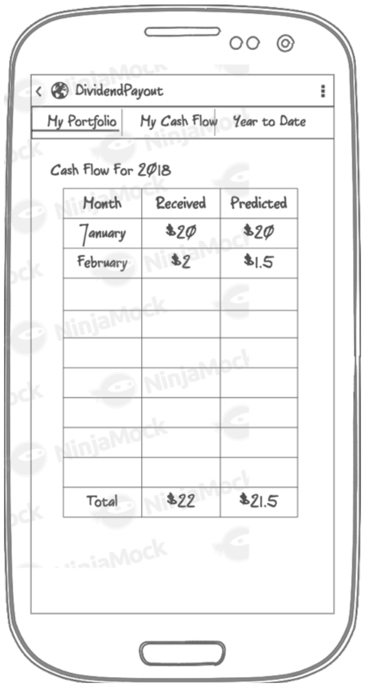


Last Upated: September 22nd, 2018 8AM

Ticker	Shares	Price	Yield	Amount	Gain
AAPL	10	$170	1.4%	$1700	$200
T	100	$33	5.9%	$3300	-$150

The screen will list the portfolio of the user by ticker. Information contained will be stock ticker, number of shares, prices, dividend yield, total amount and unrealized gain. Navigation is via tab clicks at the top of the screen. This page corresponds to the first (left most) tab.

Screen 2

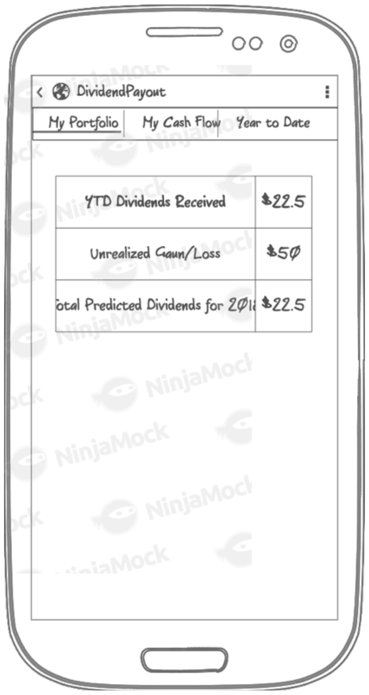


Cash Flow For 2018

Month	Received	Predicted
January	$20	$20
February	$2	$1.5
Total	$22	$21.5

The screen will contain the dividend cash flow by month for the current year. The application will predict future cash flow based on historical data. Navigation is via tab clicks at the top of the screen. This page corresponds to the second tab.

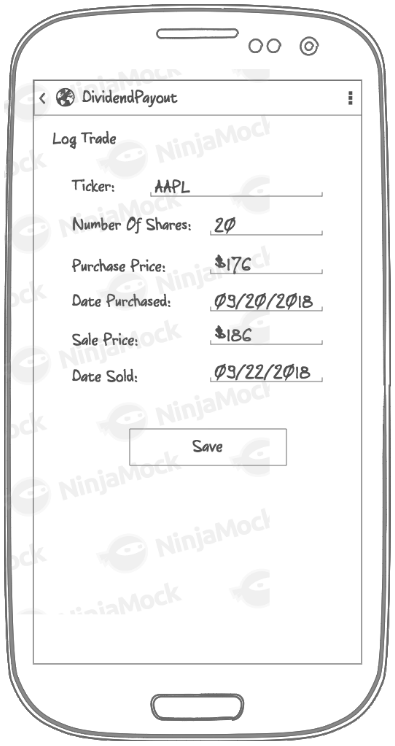
Screen 3



YTD Dividends Received	\$22.5
Unrealized Gaun/Loss	\$5Ø
otal Predicted Dividends for 2Ø1	\$22.5

The screen will contain a summary of dividends received and realized and unrealized gain for the year. The application will predict future cash flow based on historical data. Navigation is via tab clicks at the top of the screen. This page corresponds to the third (right most) tab.

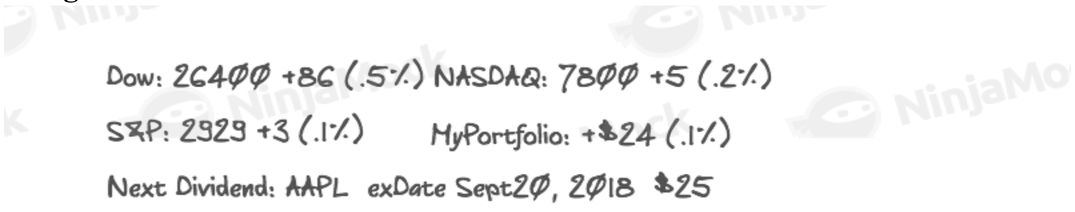
Screen 4



Ticker: AAPL
Number Of Shares: 20
Purchase Price: $176
Date Purchased: 09/20/2018
Sale Price: $186
Date Sold: 09/22/2018

The screen will be used to log trades. Information recorded will be stock ticker, number of shares, purchase price, date purchased, sale price and date of sale. This screen will be accessible by clicking on one of the stock tickers on MyPortfolio screen or by clicking on “Log Trade” item in the dropdown menu on the right side of the main toolbar.

Widget



Dow: 26400 +86 (.5%) NASDAQ: 7800 +5 (.2%)
S&P: 2929 +3 (.1%) MyPortfolio: +$24 (.1%)
Next Dividend: AAPL exDate Sept20, 2018 $25

Widget will contain key stock market indicators, most recent gain/loss for portfolio and portfolio’s next dividend payment.

Key Considerations

How will your app handle data persistence?

App will use Room persistence library to save user portfolio data.

Describe any edge or corner cases in the UX.

The user will navigate between screens by clicking on one of the tabs, by clicking on "Log Trade" on the main menu or by clicking on one of the stock tickers. Each screen will contain a back button.

Describe any libraries you'll be using and share your reasoning for including them.

Room 1.1.1 as a persistence library
LiveData 1.1.1. to hold stock data
ViewModel 1.1.1 to contain database instance
Volley 1.1.0 to consume stock API
RecyclerView V7:27.1.1 to display list of stock ticker in a portfolio
CardView V7:27.1.1 to represent each stock ticker.
Gradle 3.1.2 as a build tool

Describe how you will implement Google Play Services or other external services.

Application will use Google Ads
Application will consume an external stock API.
Application will use Google Analytics to measure user activity on each screen.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Finalize stock API selection
- Setup skeleton project containing required libraries

Task 2: Implement UI for Each Activity and Fragment

- Build UI for My Portfolio page
- Build UI for My Cash Flow page
- Build UI for YTD page
- Build UI for Log Trade page

Task 3: Initial API Integration

- Obtain necessary credentials for stock API
- Create POJOs to model API response
- Create classes necessary for API call
- Test API data retrieval
- Use Volley library to place network calls asynchronously.
- Use a SyncAdapter to pull stock data to periodically pull data from the web service.

Task 4: Implement Persistence

- Create Entity classes to represent the data model.
- Create database class
- Create DAO code for data access

Task 5: Implement Google Play Services

- Use Google AdMob to display banners at the bottom of the screen.
- Add Google Analytics to measure user activity on each screen.

Task 6: Implement Error Handling

- Display a notification when connectivity is lost to the API
- Display data that was previously stored in the database with visual indication that the displayed data is not live.

Task 7: Test and Final Build

- Execute test cases to validate application functionality
- Build and submit project for review.

Add as many tasks as you need to complete your app.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"