Project: Investing for All

Umesh Parmar 2018CS50424
Harshavardhan Baheti 2018CS50407
Dipanshu Patidar 2018CS50405

October 12, 2022

1 Introduction

Investments in stocks , over a long term, has proven to give consistent returns . The money we earn needs to earn for us too. And one of the avenues to make the money earn for us is the stock market, provided we play our game well, with patience and carefully.

Understanding the concepts of investments builds a very important habit - that of saving first and then spending. . . unlike most people who do the reverse. so yes ..every one should understand the concepts of savings/investments in all the avenues.

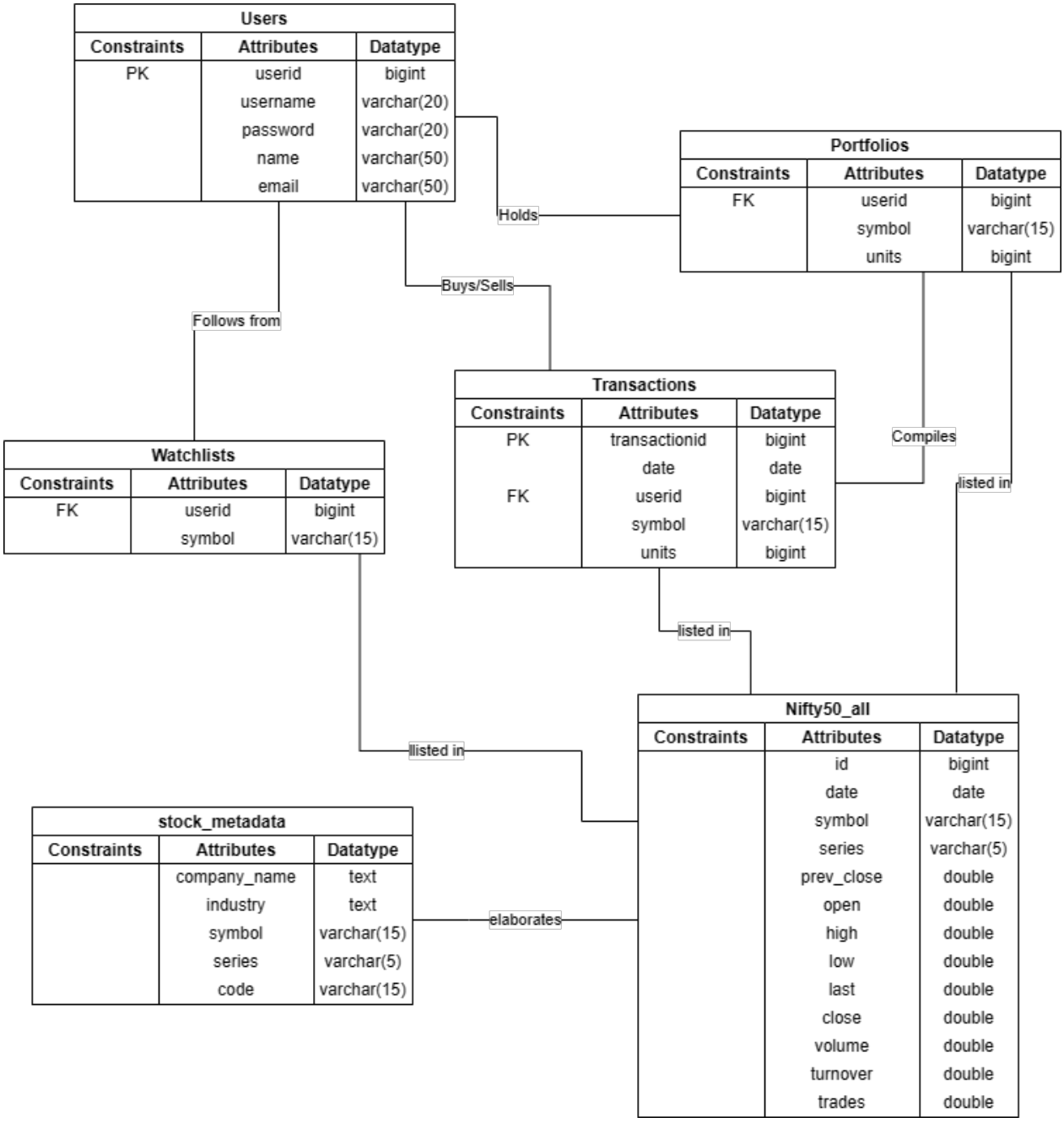
Stocks have been the most profitable asset class than any other asset class in India. Stock returns easily beat the FD returns, Gold returns, and even real estate returns in the long term. (10–20 years)

Therefore to earn better returns, to be a part of economy and earn from growing companies,to make oneself financially stable in the future, and to generate inflation beating returns, everyone should know the basics of the stock market.

To learn this basics, a simulator of the stock market is very useful tool. Therefore we chose this project, this helped all three of us to learn many things about Indian stock market. Also we learnt about the working of frontend and backend of websites.

This project simulates the Indian stock market. We consider the top 50 stocks of the Indian stock market, for this we use nifty 50 data available at Kaggle. In this data, we have tables for individual stocks containing fields like symbol, date, series, open, close, high, low, volume, etc. We also make users, which will have their own profile. These profiles will include watchlist, the current portfolio of the user(in graphical as well as tabular form), and the history of his/her portfolio. These portfolios can be updated by buying or selling the stocks. We also draw interactive graphs of all the individual stocks, which show open, close, high, low on all days along with the 52 weeks high and low.

The ER Diagram of the model can be seen below



2 Data Cleaning

1. We used data available at https://www.kaggle.com/rohanrao/nifty50-stock-market-data
2. We downloaded the data available at the above link.
3. Then we did the following cleanups:
   a)Indian stock market opens 5 days a week, therefore some dates were missing in the data, so we inserted this dates with same data as the on previous working day available.(padding)
   b) Some stock symbols were different from the stock name, so removed this by making stock symbol equal to the stock name.

3 Features and Implementation

User's view of the system

Home page

1. Initially the user sees the home page, which is accessible with/without login, where the user sees today's nifty index, the list of top-10 stocks of the on the basis of change in value in last 10 days.
2. The user also has an option to search any stocks on top right(search bar) which redirects the user to stock page of that stock.
3. Also the guest_user will be redirected to login after clicking on any other button on the the home page.

Login/register page.

1. On the login page the user has the option to login as well as sign-up(Register).
2. In login option the user can login with their username and password, the back-end will verify the user and throw an error message in case of invalid credentials.
3. In register page the user provides their name, username(need to be be unique, system will show error message in case username is already taken), password, email. The back-end will register the user and redirect to login page.
4. After successfully logging in, a session will be created for the user and the user will be redirected to their profile page.

Profile page.

1. On this page page the user can see his portfolio.
2. The user can view the current market value of their entire portfolio, their total holding in a graphical or tabular format and also graphs of their historical holdings for up to last 365 days.
3. On the right, the user can see their 'watchlist' (i.e. the stocks which the user is following).
4. The "+" button below watchlist will will redirect the user to add_to_watchlist page where user can add any stock to watchlist to follow it.
5. on the top bar the user can see his username, also it has hyperlink for home page, logout and buy/sell stock page.

stock page

1. on this page the user can see the complete details about the stock.
2. this page is accessible with/without login.
3. user can the stocks performance in any date range(manually).
4. at the bottom the user have option to buy/sell the stocks, or add the watchlist. if user is not login then this links will redirect to the login.
5. In the top bar the user has links to home page, profile page(redirect to login if not login), login page(if not login) or logout(if login)

buy/sell or add to watchlist page.

1. here the user will have enter the name of stock which he/she wants to buy/sell or add to watchlist page.
2. this page also display the list of all 50 stocks
3. then the user has 2 buttens for buy/sell and add to watchlist respectively.

4. if user clicks on add to watchlist then backend will first check for the validity of stockname (shows error message in case of invalid stockname). and then will add the stock to user's watchlist and redirect user to profile page.

5. if user clicks on buy/sell then it will redirected to another page where user has to the number of units user want to buy/sell.

System view:

1. Created views on profile page to speed up repeated data accesses.

List of all queries:

1. some small and trivial query are not included here. all these queries can also be found in app.py file.

2. To get the top stocks based on change in value in last 10 days

```
select f1.symbol, f2.close-f1.close as change,
((f2.close-f1.close)/f1.close)*100 as perchange
from
(select symbol,Date,close
from nifty50\textunderscore all
where Date='{}') as f1
,(select symbol,Date,close
from nifty50\textunderscore all
where Date='{}') as f2
where f1.symbol=f2.symbol
order by change desc
limit 10;
```

3. To get the percent change in value of top stocks from yesterday to today

```
select f1.sum, (((f1.sum-f2.sum)/f1.sum)*100) as perchange
from
(select Date,sum(close)
from nifty50_all
group by Date) as f1,
(select Date,sum(close)
from nifty50_all
group by Date) as f2
where f1.Date='{}' and f2.Date='{}'
```

4. To verify and fetch the stock from stock_metadata

```
select * from stock_metadata where symbol='{st.upper()}'
```

5. to get the the performance of given stock on all dates for ploting graph.

```
SELECT * from "+stockname+"
where symbol = '{stockname}' and Date <= '{current date}'
```

6. to get the today's performance of the stock

```
SELECT * from "+stockname+"
where symbol = '{stockname}' and Date = '{current date}'
```

7. to get 52week highest performance of the stock

```
SELECT symbol,max(high) from "+stockname+"
where symbol = '{stockname}' and Date <= '{current date}'
and Date > '{52 week before date}' group by symbol
```

8. to get 52week lowest performance of the stock

```
SELECT symbol,min(low) from "+stockname+"
where symbol = '{stockname}' and Date <= '{current date}'
and Date > '{52 week before date}' group by symbol
```

9. To verify user credentials

```
SELECT * from users
where username = '{}' and password = '{}'
```

10. To update user table

```
INSERT INTO users VALUES((select count(*)+1 from users),
'{}','{}','{}','{}')
```

11. To update user table

```
insert into transactions (transactionid,date,userid,symbol,units)
VALUES({transactionid+1},'{date}',{g.user.id},'{stockname.upper()}',{quantity})
```

12. To find daily updates in all stocks in user's watchlist

```
with todayVals as (
    select w.symbol, n.close
    from watchlists w, nifty50_all n
    where n.date = {date}
    and w.userid = {g.user.id}
    and w.symbol = n.symbol
), yestVals as (
    select w.symbol, n.close
    from watchlists w, nifty50_all n
    where n.date = \'\{date_yesterday\}\'
    and w.userid = \{g.user.id\}
    and w.symbol = n.symbol
)
 select t.symbol,
        t.close,
        (t.close-y.close)*100.0/y.close as change
 from todayVals t, yestVals y
 where t.symbol = y.symbol
```

13. To create view of user's holdings

```
create or replace view holdings_{g.user.id}_{date.strftime(dateformat)}
as
select p.symbol,
       (n.close * p.units) as value,
       p.units,
       n.close
from  portfolios p, nifty50_all n
where n.date = \'{date}\'
and p.userid = {g.user.id}
and p.symbol = n.symbol
```

14. To find the value of user's portfolio and daily change

```
with yestVals as (
  select p.symbol,
         (n.close * p.units) as value
```

```
  from  portfolios p, nifty50_all n
  where n.date = \'{date_yesterday}\'
  and p.userid = {g.user.id}
  and p.symbol = n.symbol
), todaySum as (
  select sum(h.value) as val
  from holdings_{g.user.id}_{date.strftime(dateformat)} h
), yestSum as (
  select sum(value) as val
  from yestVals)
select t.val,
       (t.val-y.val)*100/y.val as change
from todaySum t, yestSum y
```

15. To find the historical value of the user's portfolio

```
 with currentPortfolio as (
  select cast(\'{date.strftime(fstring)}\' as date) as date,
           symbol,
           units
  from holdings_{g.user.id}_{date.strftime(dateformat)}
), transactionUnits as (
  select t.date,
           t.symbol,
           sum(t.units) as units
  from transactions t
  where t.userid = {g.user.id}
  and t.date > (select distinct date-365 from currentPortfolio)
  group by t.date, t.symbol
), allDates as (
   select date,
           allSymbols.symbol
   from ongc,
       (select distinct symbol
       from transactionUnits) as allSymbols
  where date <= (select distinct date
              from currentPortfolio)
  and date > (select distinct date - 365
              from currentPortfolio)
), transactionPadded as (
  select a.date-1 as date,
       a.symbol,
       (case
           when t.units is NULL then 0
           else t.units
       end)
  from transactionUnits t
  right outer join allDates a
  on t.date = a.date
  and t.symbol = a.symbol
), transactionCumulative as (
  select t.date,
       t.symbol,
       sum(t.units)
       over (partition by t.symbol
              order by t.date desc) as units
  from transactionPadded t
), histPortfolios as (
```

```
  select t.date,
       c.symbol,
       (case
           when c.units is NULL then 0
           else c.units
       end) -
       (case
           when t.units is NULL then 0
           else t.units
       end) as units
  from currentPortfolio c
  full outer join transactionCumulative t
  on t.symbol = c.symbol
)
select h.date,
       sum(h.units*n.close) as value
from histPortfolios h, nifty50_all n
where h.date = n.date
and h.symbol = n.symbol
group by h.date
order by h.date asc
```

Query Execution time: The query running times are reported in the following table. Note: Simple queries with running time ¡10ms have been excluded.

Query No.	Execution time(including fetch)
1	437 ms
2	58 ms
11	146 ms
12	79 ms
13	142 ms
14	176 ms