TRADING RISK MANAGEMENT SYSTEM

MC-122 PROJECT
~ KAUSHAL NANDANIYA-202303036
~GANTAVYA MAYAL-202303033

User Manual Report

In our program, We are using the database using MySQL which is storing and using the database named ‘stocks’ for fetching data for different stocks which is being stored in ‘stocks’. So the user system must have this database in their system and also java database connector

Also to run the java code one should run the main code ‘App.java’ along with specifying the class path ofJ/connector of database.for example,

```
java -cp .:/Users/kaushalnandaniya/Desktop/Project/mysql-connector-j-8.3.0/mysql-connector-j-8.3.0.jar App
```

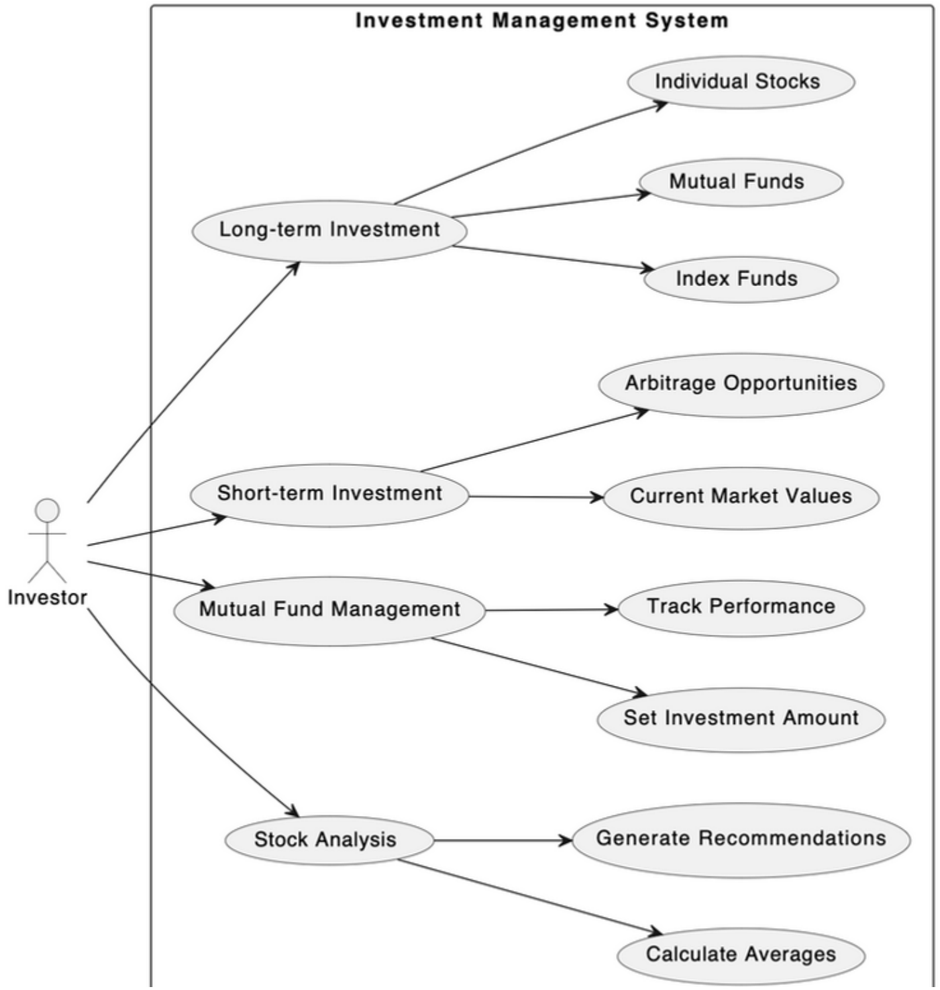
As a default the Username and Password is set as ~

Username : root

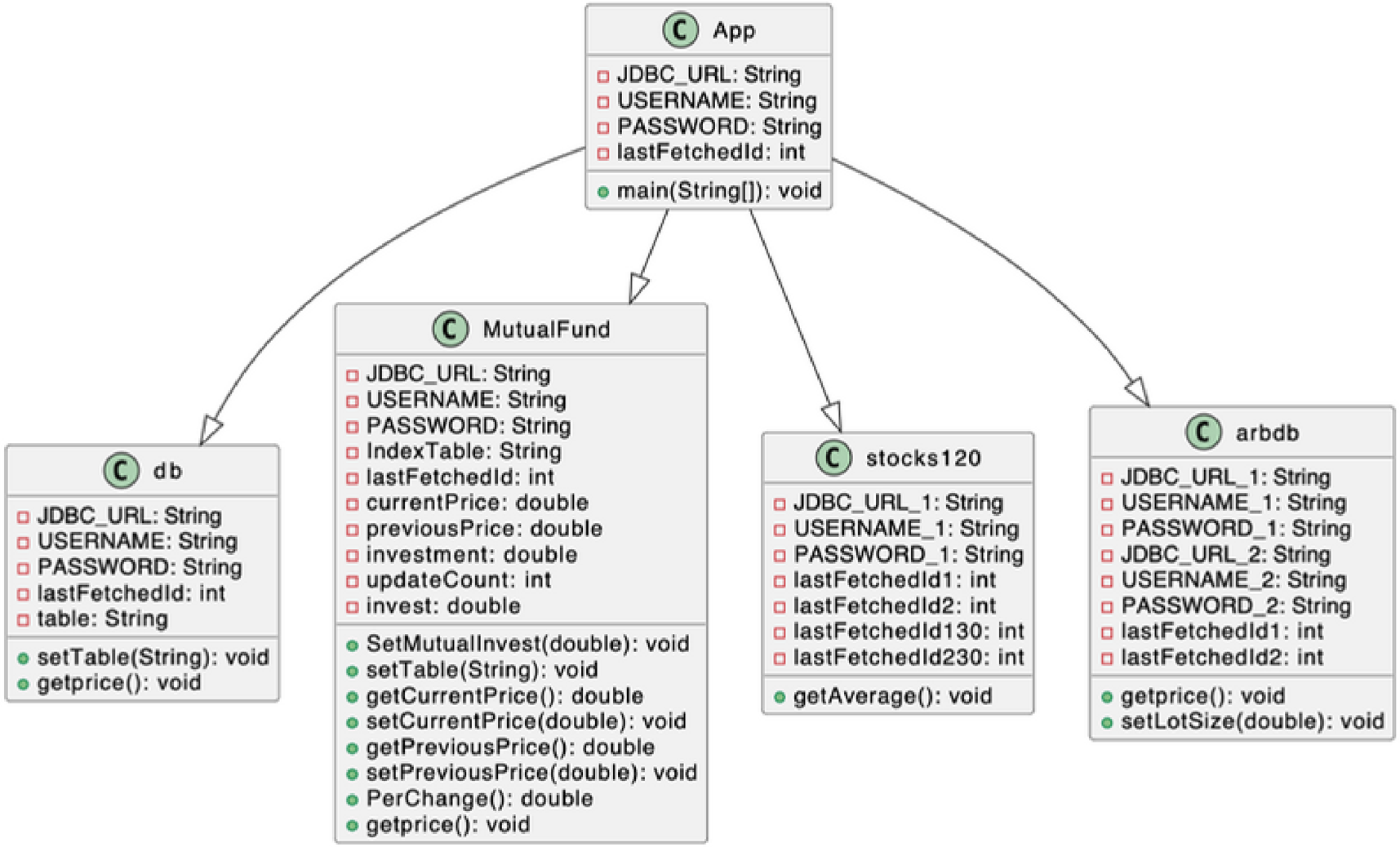
Password : Kaushal@2004

NOTE : User can replace it with their Username and Password.

UML Diagrams of our program



This is the UML diagram which depicts the functions of investor in stock market. A investor can invest in analytical stocks, mutual funds, indexes and also short term arbitrage with the help of database for analytical part.



This UML class diagram shows diagram illustrates a financial investment system with classes like **App**, **db**, **MutualFund**, **stocks120**, and **arbdb**. These classes manage various aspects such as database access, mutual fund investment, stock analysis, and arbitrage trading. Associations and dependencies between classes depict the relationships and interactions within the system.

This project utilises object-oriented programming to help user/investor to invest the money in efficient and analytical way so that he/she can increase their wealth with low risk and higher rewards.

- Upon running the program correctly as explained in user manual report, running App.java using classpath of J/connector, The code will load successfully with message “MySQL JDBC driver loaded successfully” as main method will check wether the J/connector is loaded or not.

```
try {
    Class.forName(className:"com.mysql.cj.jdbc.Driver");
    System.out.println(x:"MySQL JDBC driver loaded successfully");
} catch (ClassNotFoundException e) {
    System.err.println(x:"Failed to load MySQL JDBC driver");
    e.printStackTrace();
    return;
```

- If loaded successfully then user will be asked for long and short term investment

```
MySQL JDBC driver loaded successfully
ENTER:
1 : For the long term investment
2 : For the short term investment
```

- If User entered rather then 1 or 2 , the user will be again asked to enter the number.

```
Invalid choice.
ENTER:
1 : For the long term investment
2 : For the short term investment
```

If the user entered 1 or 2 the cod will execute accordingly since 1 will lead to longterm investment and 2 will lead to short term investment.

1) Here the user will be given options for long term investment as below

```
categories for Long term Investment
Enter
1 : For Index funds
2 : For Mutual funds
3 : For Stocks
```

Further the algorithm will run accordingly as long term investment option is selected.

2) Here the user be given option to see the current value of the market or can do the Arbitrage Opportunities Trading

```
ENTER:
1 : For the current value
2 : For the Arbitrage Opportunities Trading
```

This is the over view of the code. we will see in detail for every investment options and
the algorithm and methods behind it.

Class long_term

This class will act as super and has a lonterm( ) method which help the child class to perform and get the current price of the selected stocks

```
void lonterm() {
    System.out.println(x:"ENTER for current price:");
    Scanner stable = new Scanner(System.in);
    String st = stable.nextLine();
    String UpperCase = st.toUpperCase();
    db mydb = new db();
    mydb.setTable(UpperCase);
    mydb.getprice();
    stable.close();
}
```

This lonterm ( ) method will set the name of the stock in “db” and will call the getprice( ) which will give current price of selected stock to user.

Class IndexFund

This class is the child class of long_term which has a constructor which give options of indexes and a MutualIndex( ) method which will set the Index and invest in it through MutualFund public class.

```
id MutualIndex() {
  System.out.println(x:"Enter any of the Index:");
  Scanner stable = new Scanner(System.in);
  String st = stable.nextLine();
  String UpperCase = st.toUpperCase();
  MutualFund mutind = new MutualFund();
  mutind.setTable(UpperCase);
  System.out.println(x:"Enter the amount to invest");
  Scanner sinvestIndex = new Scanner(System.in);
  double investIndex = sinvestIndex.nextDouble();
  mutind.SetMutualInvest(investIndex);
  mutind.getprice();
  sinvestIndex.close();
  stable.close();
```

Class MutualFuc

This is also the child class of long_term, it contains Mutualfu( ) as a method which will call public class MutualFund and set the amount to be invest in it which will automatically add the amount after 30 day as done in mutual fund

```
void MutualFu() {
    Scanner scanner = new Scanner(System.in);
    System.out.println(x:"Enter the amount to invest:");

    double n = scanner.nextDouble();
    MutualFund mutualFun = new MutualFund();
    mutualFun.SetMutualInvest(n);

    scanner.close();
    mutualFun.getprice();
}
```

If we do so the follwing will happen

```
Enter the amount to invest:
1000
Updated Investment: 1000.000
Updated Investment: 1040.599
Updated Investment: 1021.093
```

```
Updated Investment: 1083.717
Updated Investment: 1095.541
valuation after 1 months: 2095.5406891717876
Updated Investment: 2109.914
```

public class stocks120

In long term the investment through stocks is done by public class stocks120 which will calculate 30 and 120 past days average from current day, helps the user to predict the market. If 30 days average is greater then 120 days average then market is going in the up trend and algo will suggest to BUY the stock and for vice versa to SELL the stock.

```
Average120: 148.57708322499997
Average30: 175.39499870000006
BUY
```

```
Average120: 157.92291624166663
Average30: 152.19499946666667
SELL
```

class short_term

short_term class has a constructor will assign the options for short term investment and a ArbitrageIndex() method which will set the lot size and call the getprice() from public class “arbdb”.

```
void ArbitrageIndex() {
    arbdb arbdb = new arbdb();
    System.out.println(x:"Enter the Lot Size :");
    Scanner slot = new Scanner(System.in);
    int n = slot.nextInt();
    arbdb.setLotSize((double) n);
    slot.close();
    arbdb.getprice();
}
```

The public class arbdb will get instance price of the a stock from two different stock market and compare it, if there is any price difference in it then it will buy the stock from market which has low price and sell it to market which has higher price at that instance. Also difference will allocate to lot size and add to profit.

```
open: 182.050003, close: 181.350006
open: 181.75, close: 181.350006
Profit: 9799.995499999994
```

THANK YOU

PRICE IS WHAT YOU PAY,
VALUE IS WHAT YOU GET.