**1** Introduction

* 1. **Company Profile**



Pirimid Fintech LLP founded by Nirav Prajapati in 2017. Pirimid Fintech LLP is a Limited Liability Partnership firm incorporated on 17 October 2017.

Pirimid Fintech provides cutting edge software development services with highest quality, robust software systems and workflows. Pirimid have expertise in capital markets across various asset classes building execution & order management, low latency trading, risk & compliance management, business intelligence, algorithmic trading, crypto currency trading, open banking, account aggregation, digital lending, blockchain and machine learning. The company’s segments are Financial services and insurance, Capital markets along with order execution management systems, Low latency, High frequency, REST/FIX API Trading, Risk and compliance, Algorithmic Trading, Big data and analytics, matching engine, dark pool, ECN. It provides solutions for Banking domain Early warning system and account aggregation. Company’s capital market service includes Hedge fund and Robo advisory solutions. Machine learning & Artificial intelligence service includes Pretrained Machine Learning APIs, Deep Learning & Predictive Analytics, Stock Directional Networks. Blockchain & Crypto includes services of Business process automation, KYC & Identity Utility, Hyperledger Framework.

Pirimid has been an immensely growing Company in the financial domain in recent time. Company has clients from various countries and is expanding its area to other locations also. Pirimid’s center is located at Ahmedabad at Gujarat, where we are fortunate to get trained and exposed to industrial standards.

**2**

About the system

**2.1. Problem Statement**

Traders and investors are willing to invest their money into many types of assets in day-to-day life. They are required to monitor their investments and execute orders based on their requirement. Online trading platforms will make it easier to trade using such tools and execute orders, monitor positions and watchlist.

**2.2. Aim**

To provide a straightforward solution to the above problem statement with good user interface and proper analysis of past fluctuations and movement of stock represented by graphical interface where traders can buy and sell shares with an easy method. Chart can be of multiple types e.g., candlestick chart, line chart.

**2.3 Features of system**

**2.3.1 User Registration**

New users would have to register first to use the pirimid trading platform in which they would have to provide valid information to proceed further.

Information required for registration:

1. First name
2. Last name
3. Email
4. Password
5. Date of birth

**2.3.2 User Login**

After registration users would need to login to use the system. At any point of time users can update their profile.

Login credentials:

1. Email
2. Password

**2.3.3 KYC verification**

Before doing any investment, users can go through the KYC process, where eSignature, PAN card photo and photo of user will be required to upload for verification.

KYC verification requirements:

1. Signature photo
2. User’s selfie size photo
3. PAN card photo

**2.3.4 Dashboard**

We provide the dashboard of the system which shows popular funds and popular stocks available in the market, also Handpick collections of both assets. It also shows the investment which has been done by the user on the dashboard.

**2.3.5 Stock List**

System provides the feature of finding all stocks which are available in the market. Users can filter them based upon requirement.

Filters for stocks:

1. Company Name
2. Market capital value
3. Closing price

**2.3.5 Mutual funds List**

It is the page where users can find all the mutual funds which are available for investment. Users can filter them based upon requirement.

Filter for mutual funds:

1. Company Name
2. Risk
3. Find size (cr)
4. Available to invest

**2.3.6 Asset Detail Page**

Here users find the details of particular stock & mutual fund with graphical representation. Which shows what this company does and who is the organizer of the company or organization. From this page one can sell or buy shares using a trading ticket. Chart shows open, close, high and low price of stock within any time period.

Chart have two types:

1. Candlestick chart
2. Line chart

**2.3.7 Watchlist**

If any user wants to watch or monitor only specific companies then watchlist then users can add those company assets to watchlist. From there users can see the open, close, high and low price for stock.

**2.3.7.1 Search in watchlist**

Users can search companies based on the name of the company inside the watchlist.

**2.3.8 Orders**

From the details page of the company traders can place an order from trading ticket. That order will be shown inside the orders page based on the type of asset (Stock/Mutual Funds). Each order has the status. Every order will be executed after some specific duration of time. The mutual funds system provides features of making SIP (systematic investment plans) orders.

**2.3.9. Order Ticket**

**2.3.9.1 Stock ticket**

Stock ticket provide various options to place an order for particular stock.

1. Product code
   1. CNC (Cash and Carry)
   2. MIS (Margin Intraday Square off)
2. Order Type
   1. Limit
   2. Market price
3. Action
   1. Buy
   2. Sell

**2.3.9.2 Mutual fund ticket**

It provides various options to place orders in mutual funds with SIP policies.

1. Frequency of SIP
   1. Weekly
   2. Monthly
   3. Yearly
2. Order type
   1. Lumpsum (amount to invest)
   2. SIP (systematic investment plans)

**2.3.10 Positions**

A position is an amount of security or assets that is owned by some individual or some entity. A trader or investor takes a position when they make a purchase through a buy order. It is a portfolio owned by individuals on their current buying quantity.

**2.4. Workflow of system**

**Steps:**

1. User request for login with credential details.
2. Server checks the database for user entry.
3. If user is present in database, then backend will generate JWT (JSON Web Token)
4. Then it sends to the client or frontend or browser.

**3**

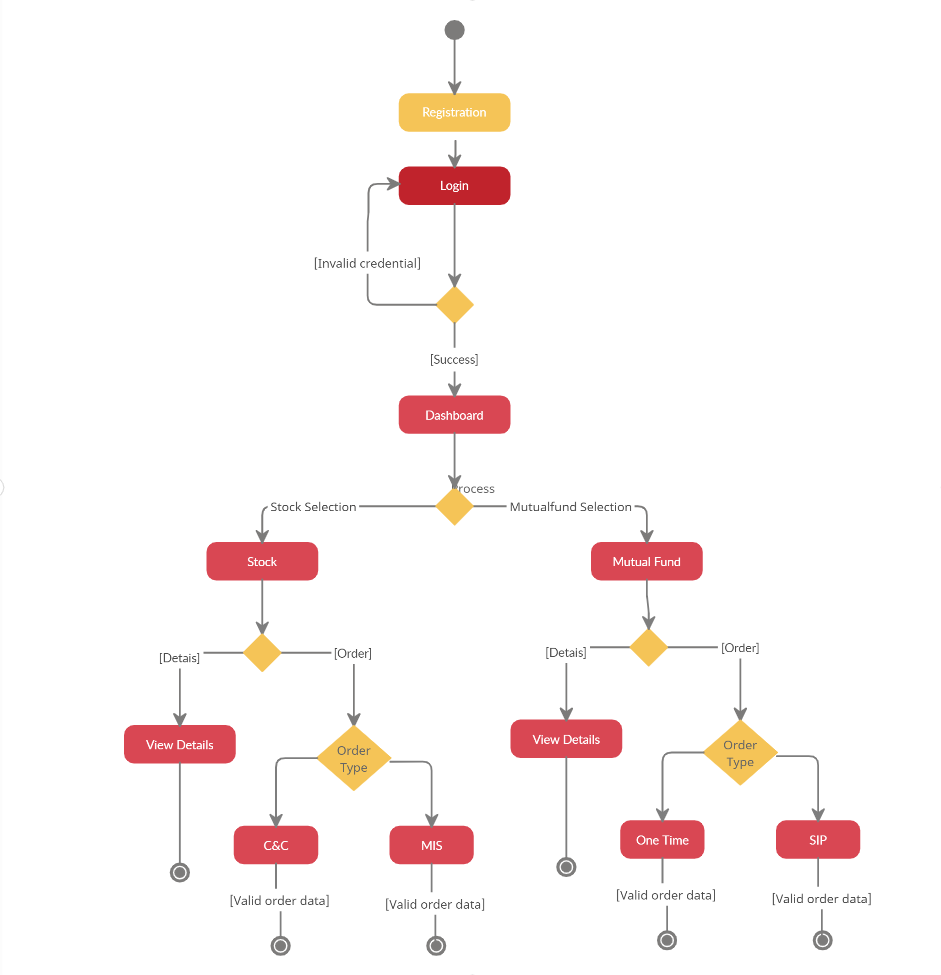
Analysis

**3. UML Diagrams**

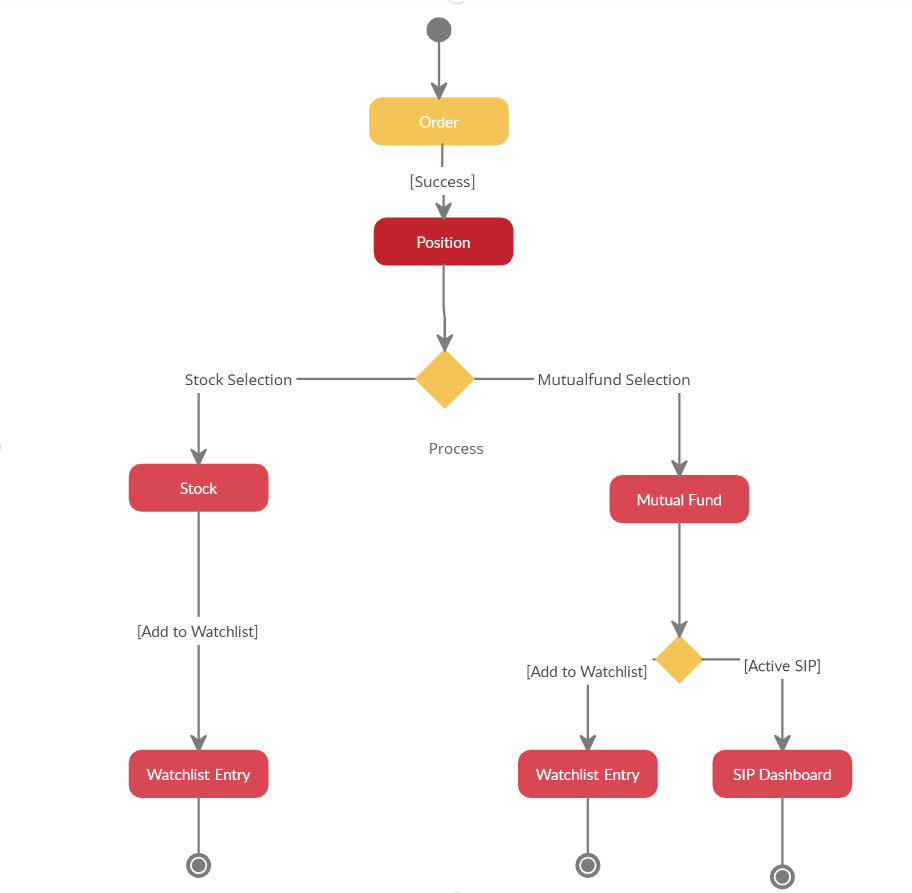
**3.1 Activity Diagram**

Activity Diagrams Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

**3.1.1 activity diagram for order**



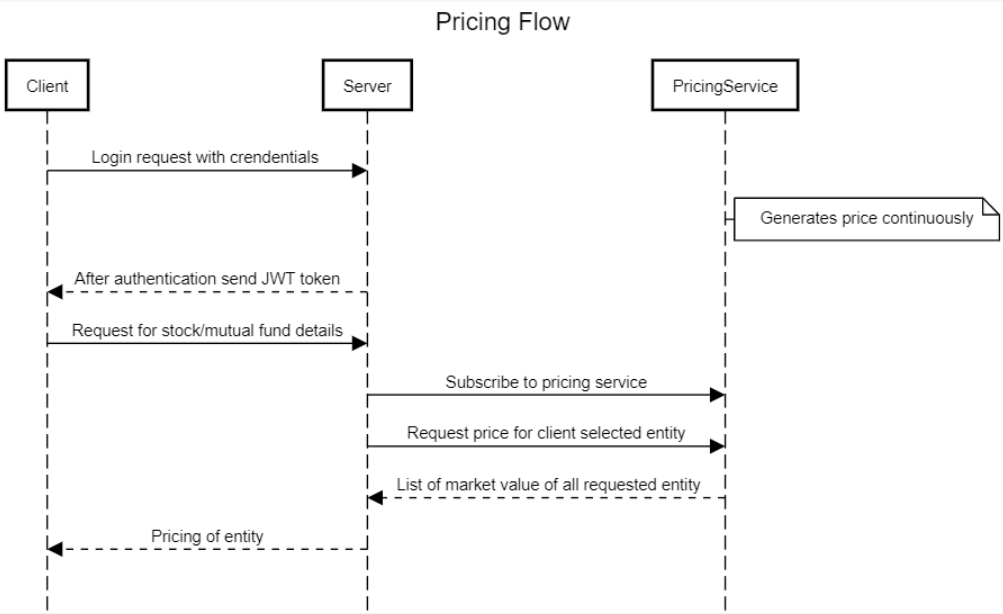
**3.1.2 Activity diagram for position**



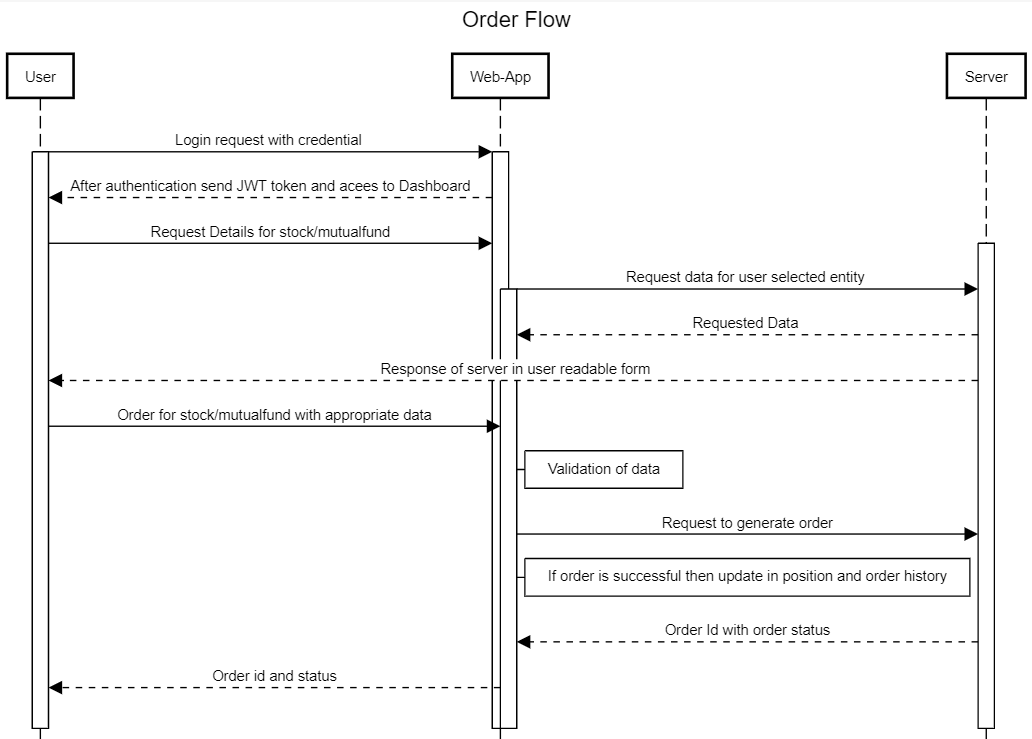
**3.2 Sequence diagram**

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

**3.2.1 Sequence diagram for pricing flow**



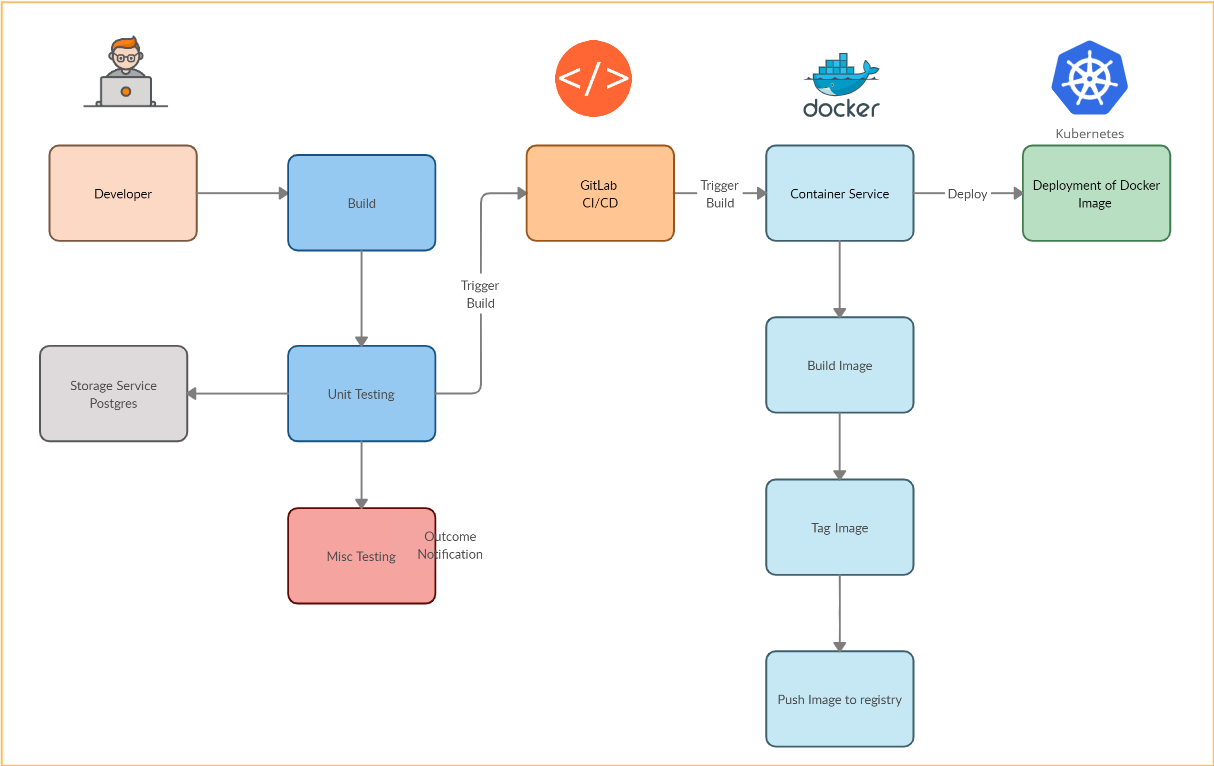
**3.2.2 Sequence diagram for order flow**



**3.3 High-Level Development Flow**

It shows high level development flow for the whole application from scratch development to deployments on kubernetes. As depicted in images first phase is of development and after building any module its tested in unit testing and after that its pushed on gitlab where CI/CD trigger the build to create an image on docker and after completion of this phase it’s deployed on kubernetes.

**3.3.1 Overall application flow in development**



**4**

**Design**

**4.1. UI Mocks**

**Pirimid Trading Platform**

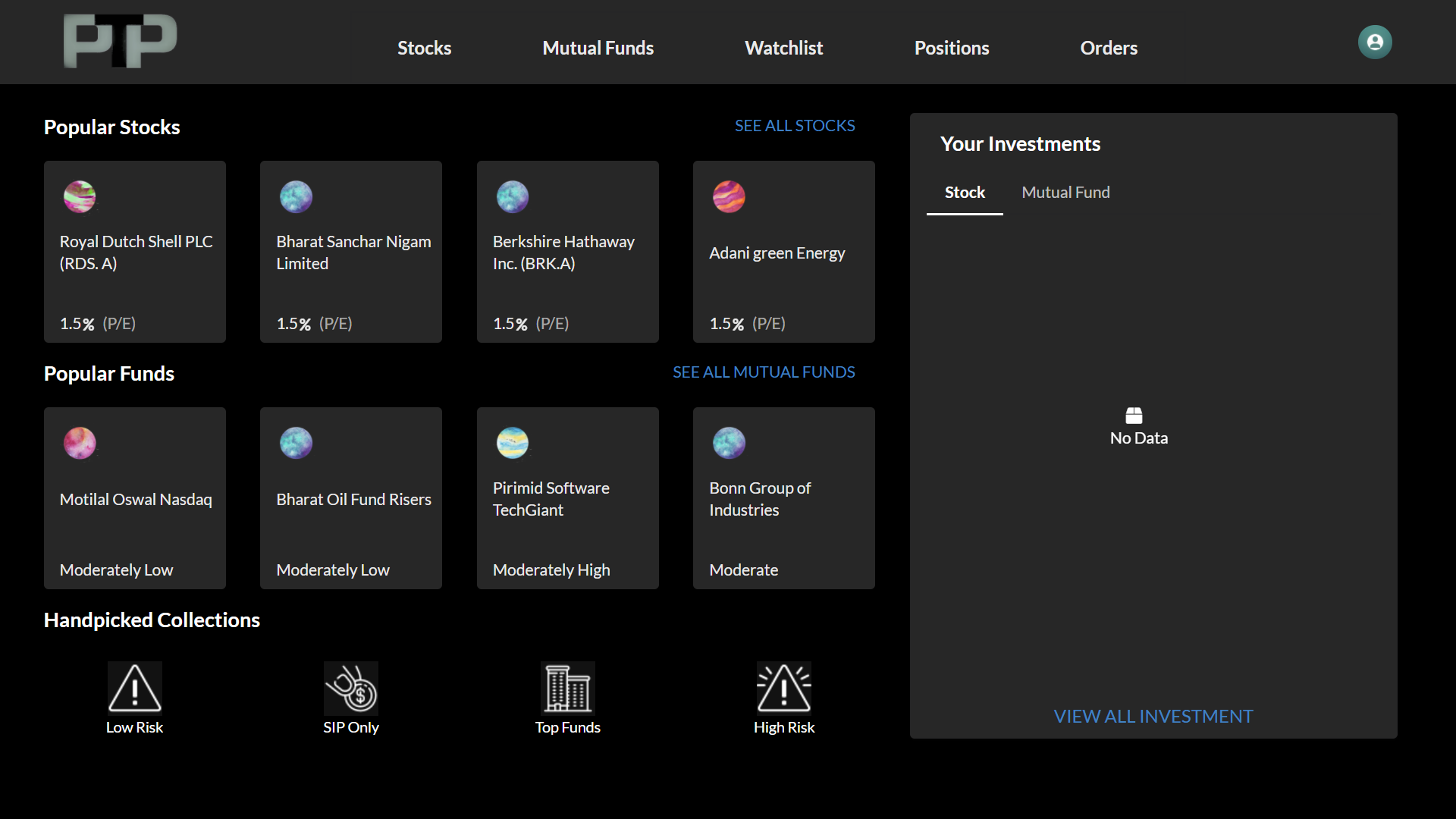


Figure4.1-Pirimid Trading Platform-Dashboard

**Authentication of user**

* Users have to go through the registration and login process, which involves simple registration with email and password.
* For further authentication of user in our session less system we are using

JWT (Json Web Token) as base for authentication

**KYC**

* Without KYC verification user can not involved in buying or selling of any asset of this system
* Users have to have KYC verified to utilize full benefit of the system.

**Dashboard**

* Figure3.1 shows the landing page for the user in the system. When we have most popular assets to invest in as well as their current standing in the share market.
* System also suggests handpicked collections for our assets, formed by experts to give amateur investors a good idea about where to start.
* Dashboard also provides navigation to respective screens for assets, portfolio management and watch over particular assets.

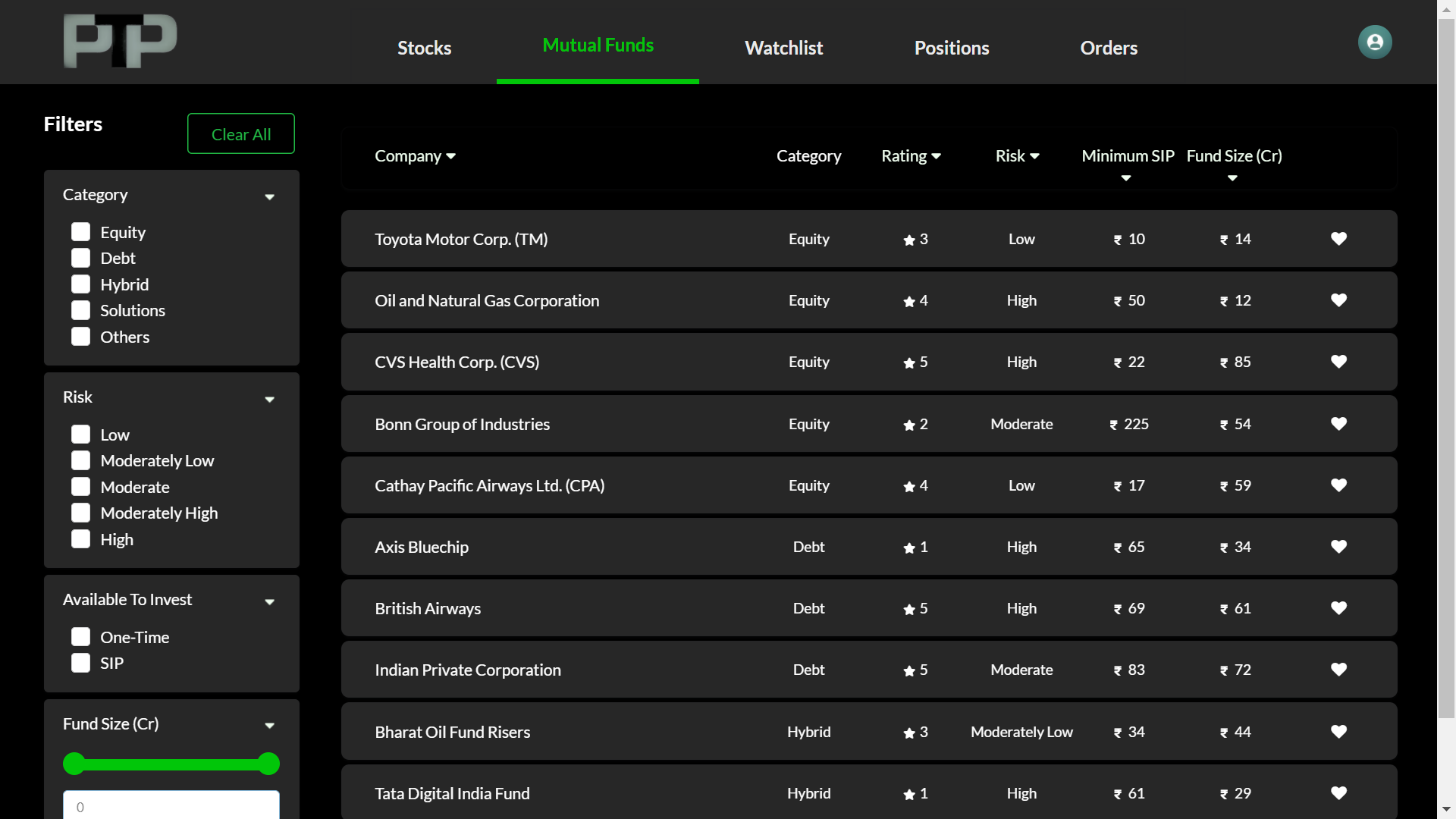


Figure3.2-mutualfunds

**Mutual Funds**

* On selecting Mutual funds from navigation, it will land on screen which looks like Figure3.2.
* On this screen we have all the Mutual funds available to invest in the system. Which also supports a variety of filters.
* Filters are meant to provide ease in searching appropriate assets to invest in. Mutual fund page also shows lots of details regarding it’s ratings, various factors related to investing to help investors evaluate the asset.
* There is also a heart shaped icon at last to add individual assets to the watchlist for quick access. For reference Figure3.2.1-add to watchlist.

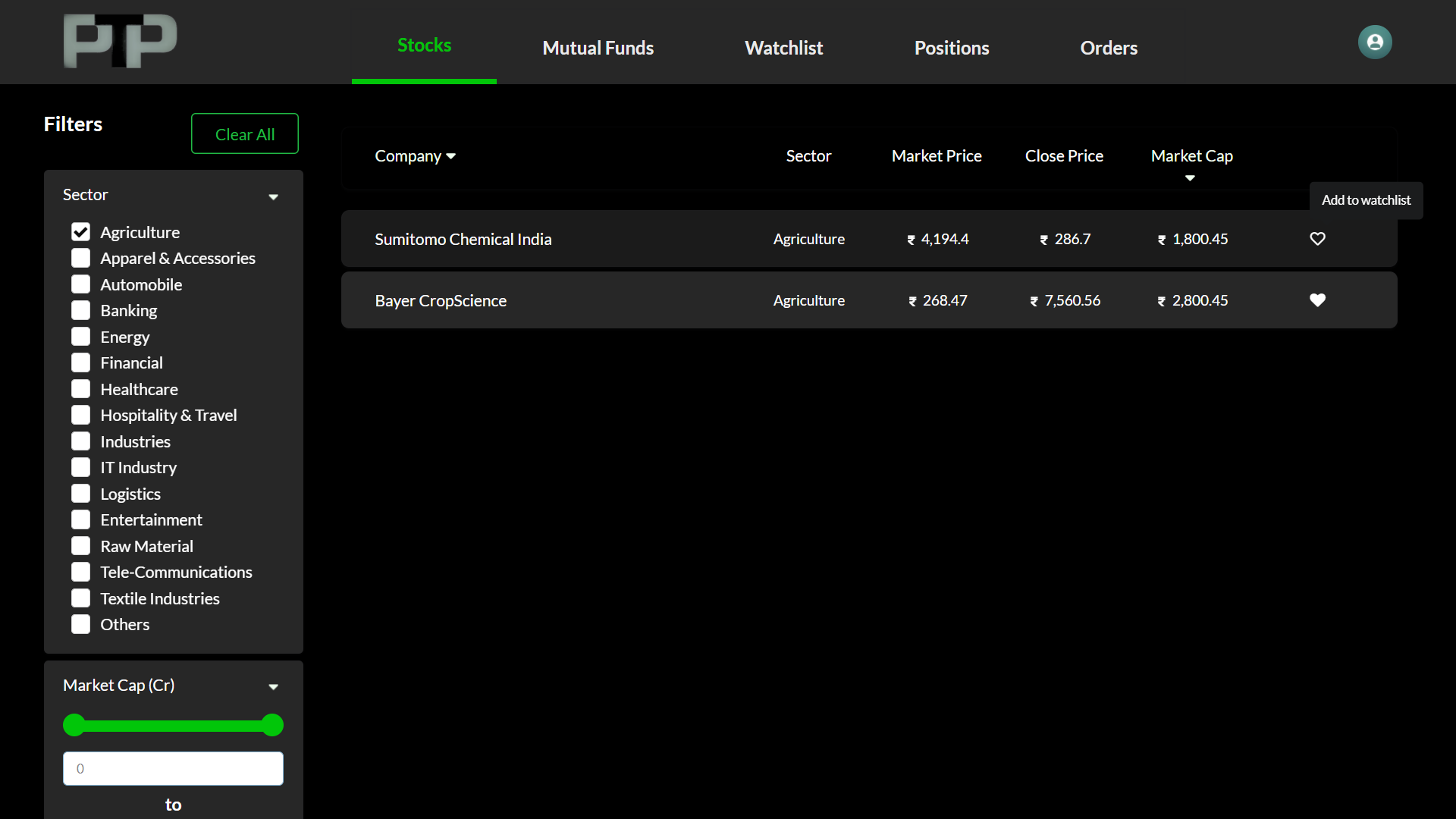


Figure3.2.1-add to watchlist

**Results of Mutual funds filters**

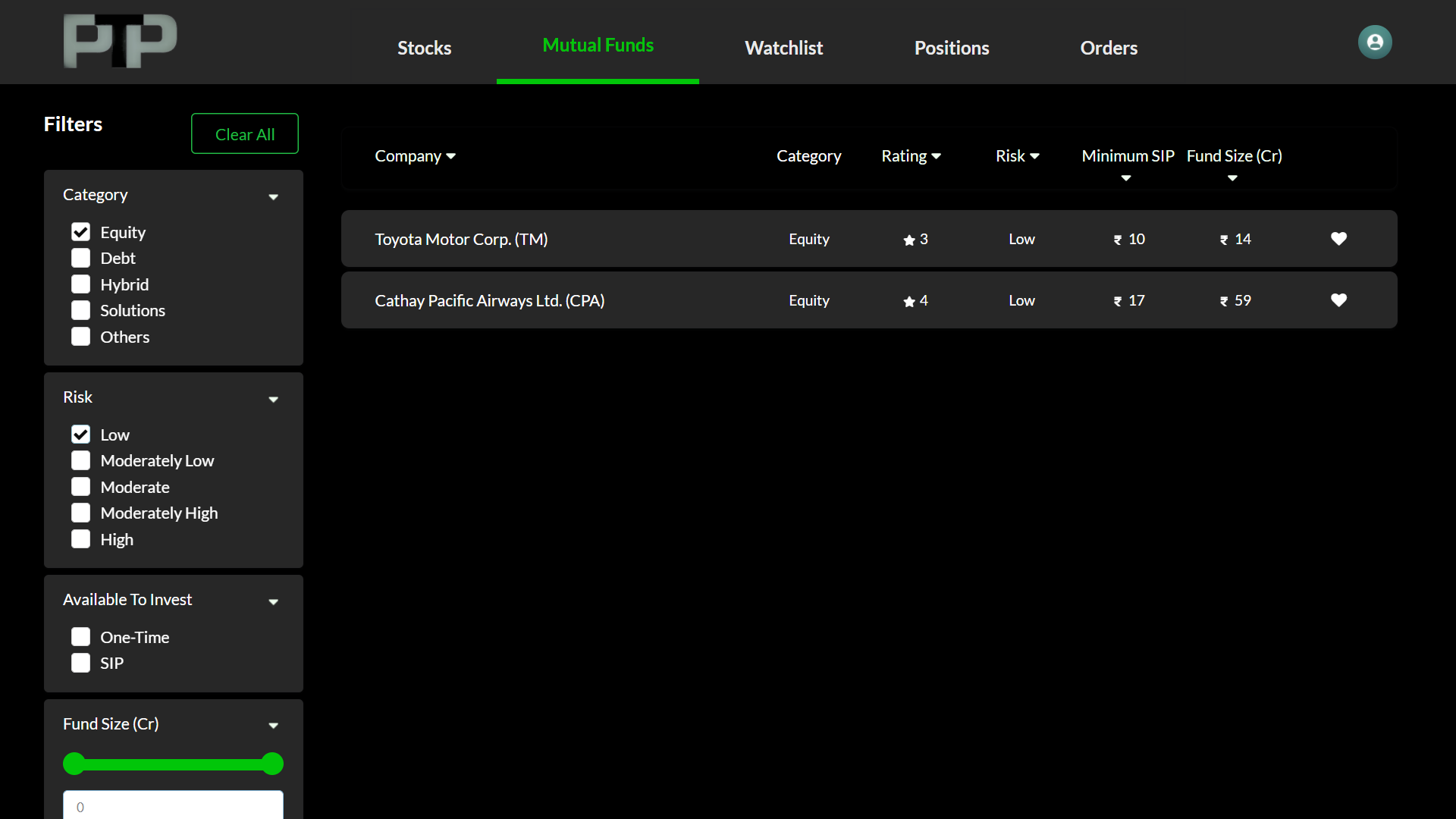
******

Figure3.3-mutual funds filtered by equity and low risk

* As shown in Figure3.3, it uses filters to get results for assets involving categories such as equity and where investment Risk is Low.
* There are also headers available to provide sorting in ascending or descending order.

**Before Start Investing**

* Users have to be a KYC verified, our system uses Pan card scanning which will be matched with id for validation.
* Figure3.4.1, Figure3.4.2 and Figure 3.4.3 show screens for validating users in the system.
* Screen 1 takes users information, Screen 2 takes pan card as an image and Screen 3 takes signature and users photo for the system.

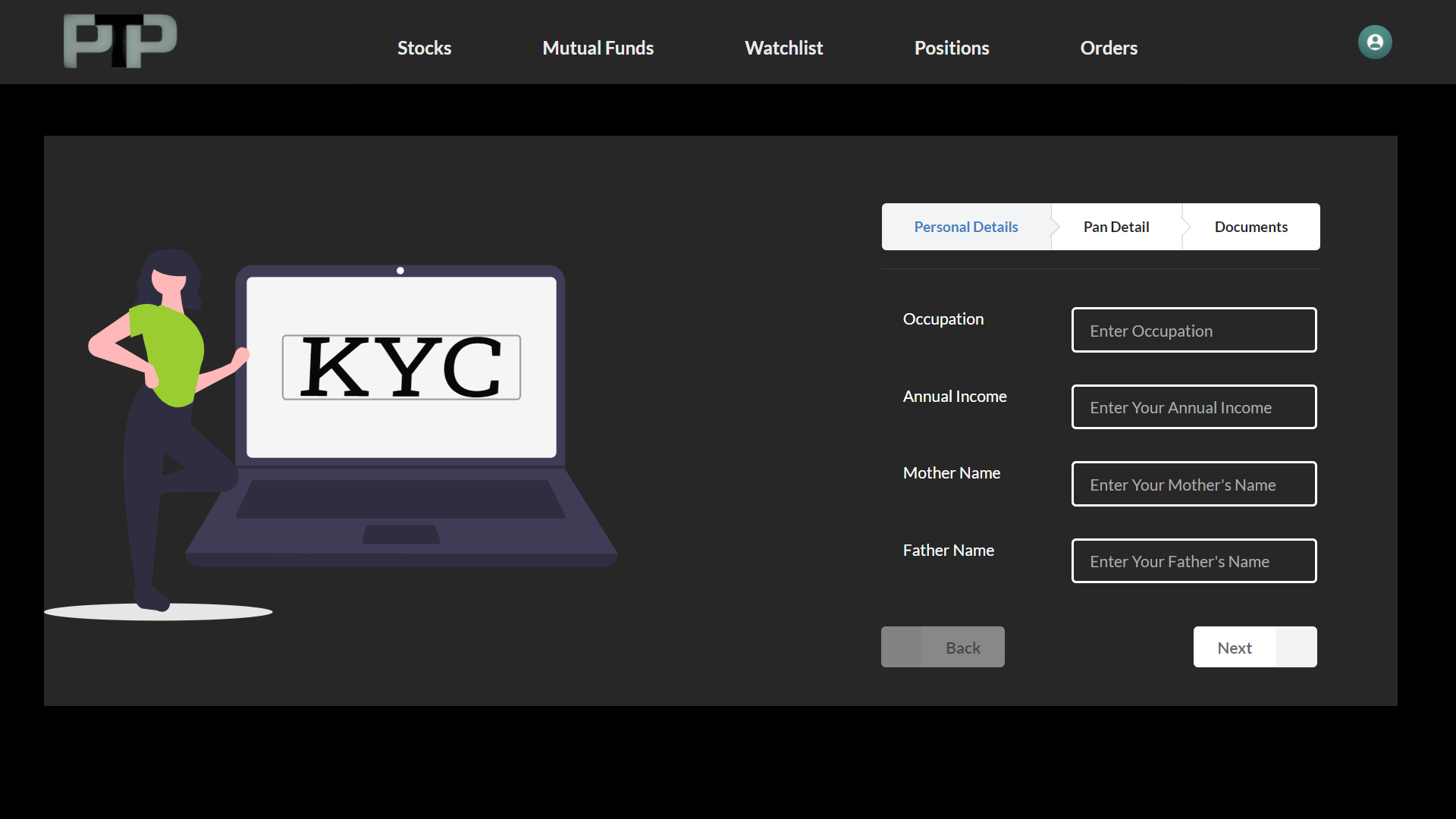
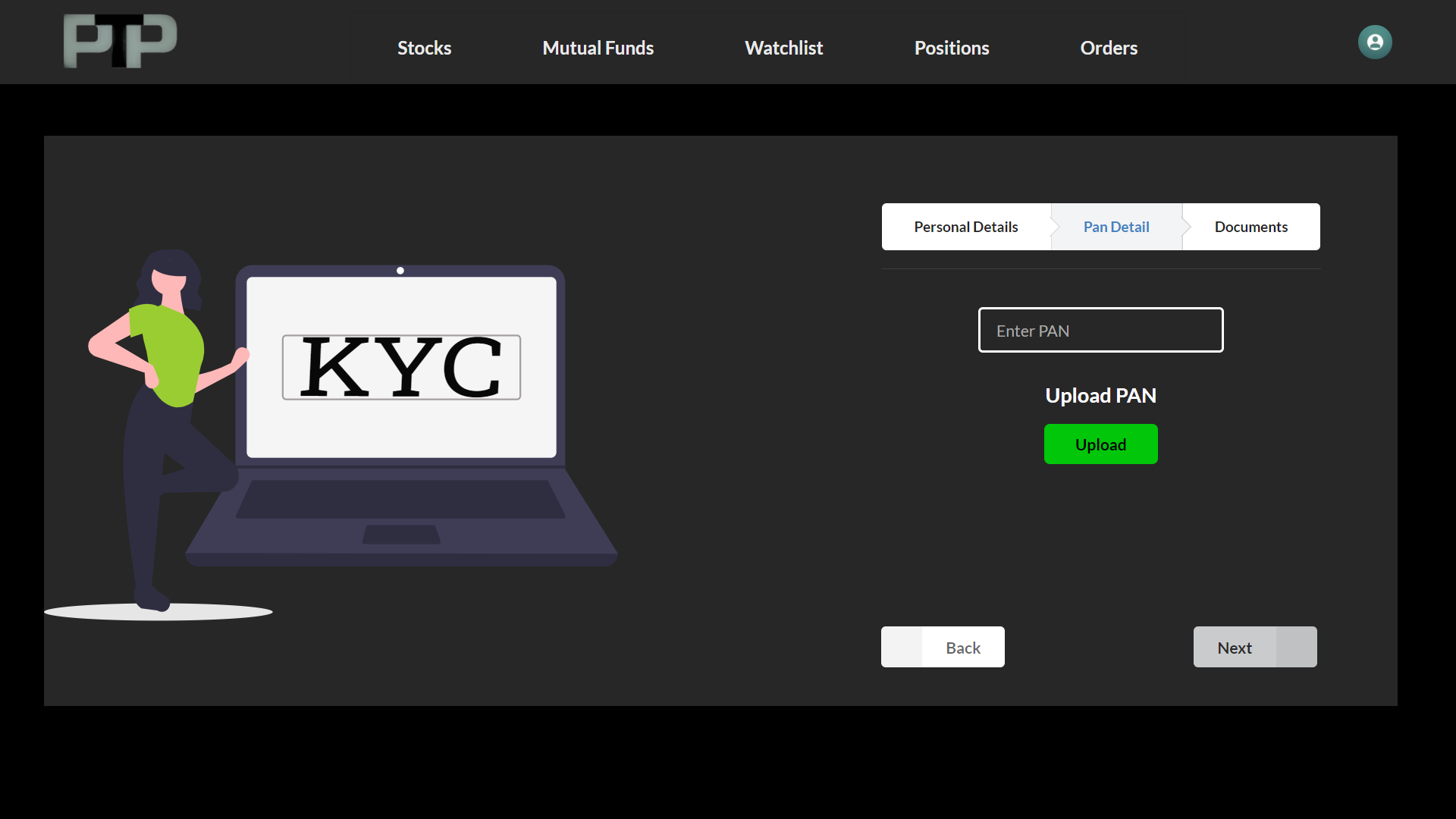


Figure3.4.1-kyc step 1



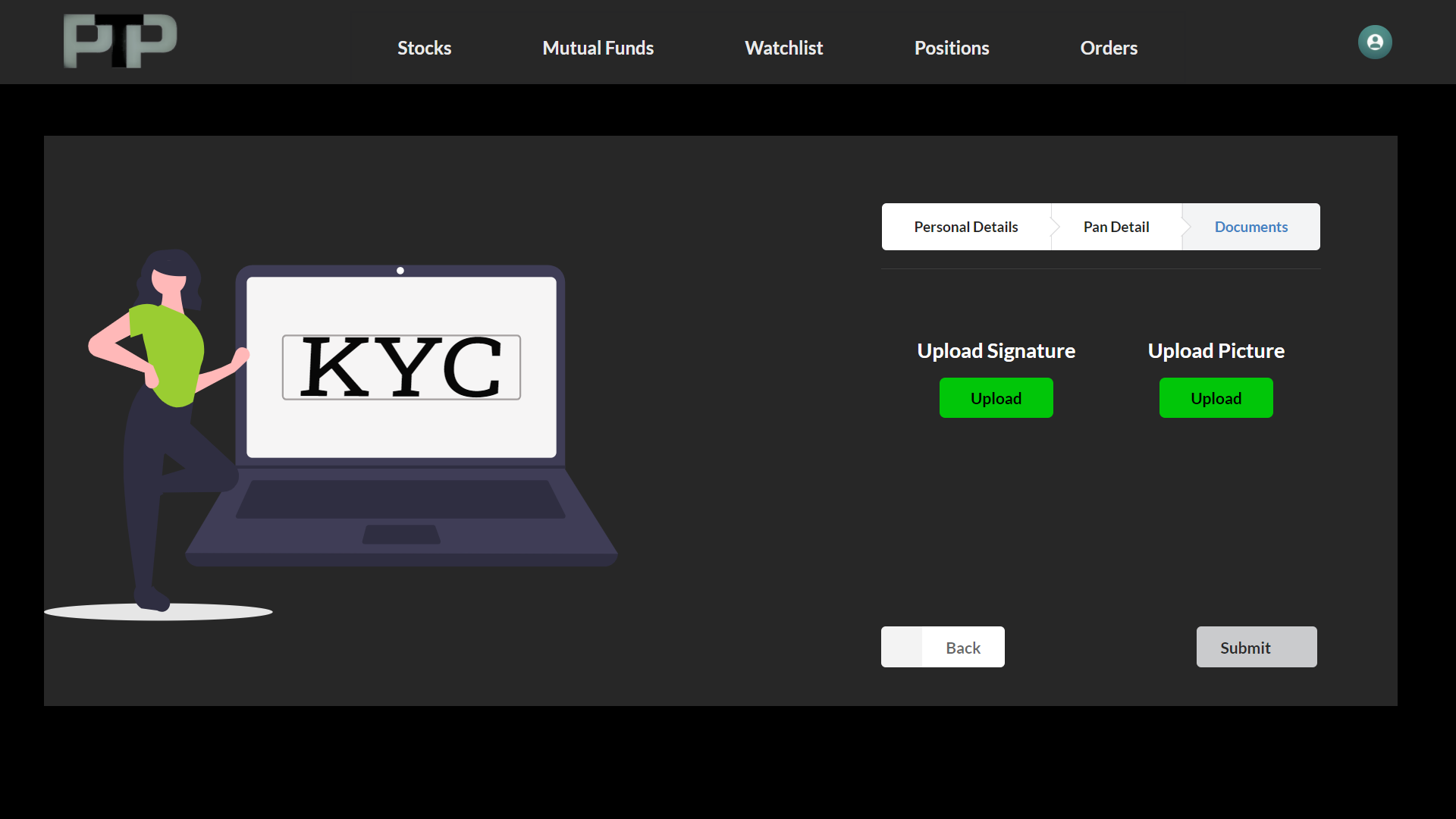
**

Figure3.4.1-kyc step 2 Figure3.4.2-kyc step 3

* Here these screens take required documents and information about users to start investing in assets. It will take a signature, current user picture and pan card.

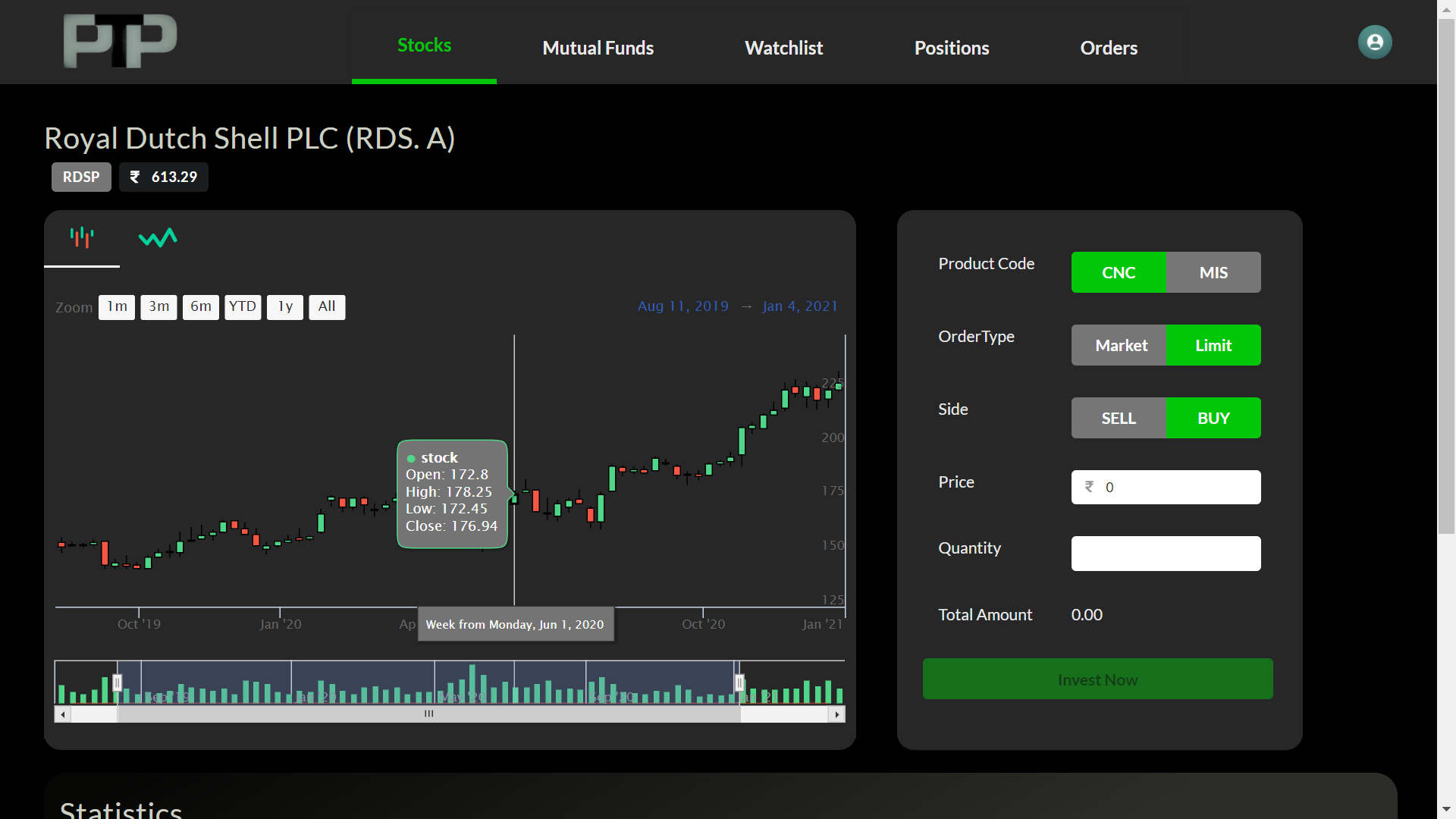
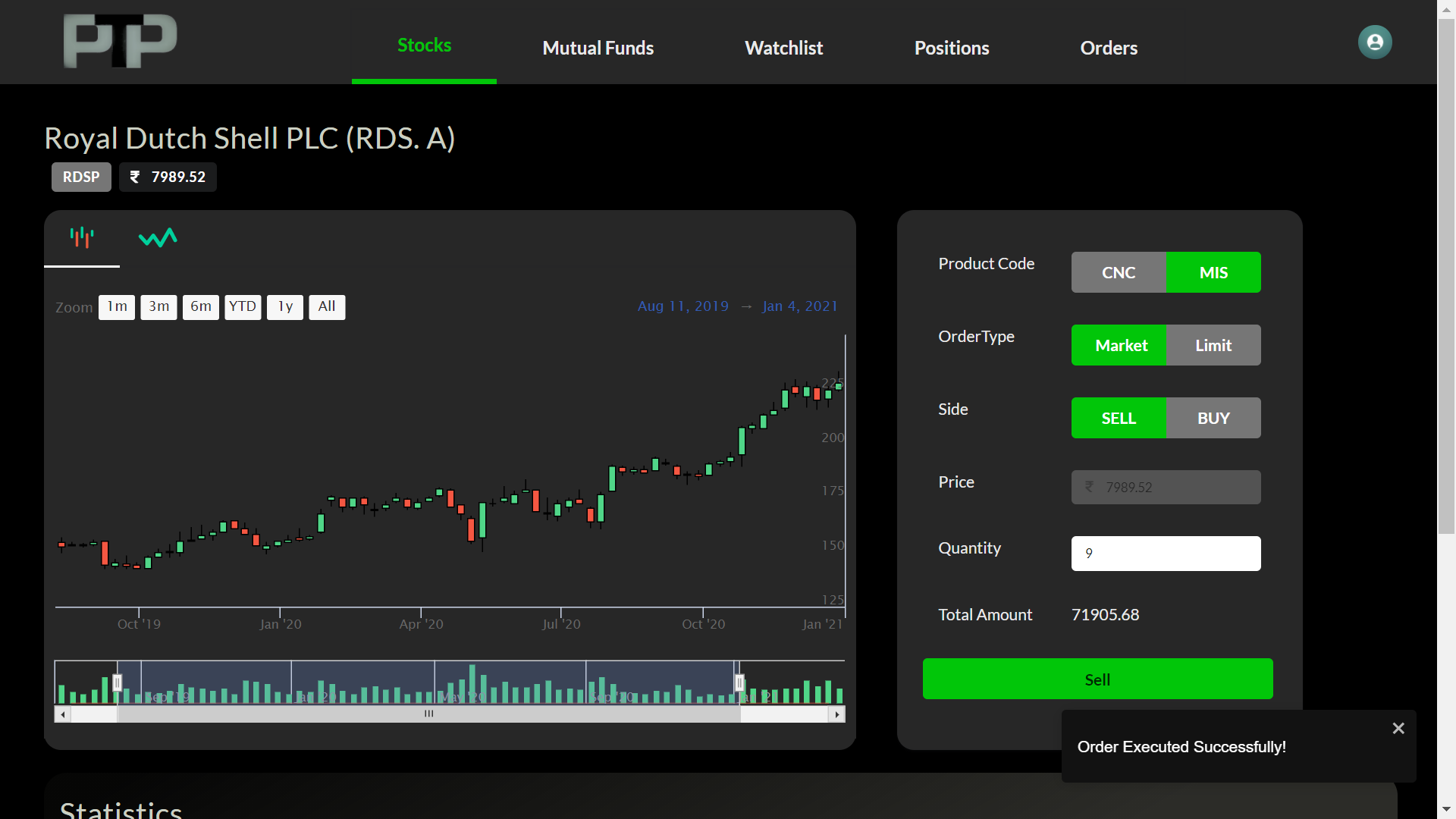
**

Figure 3.5-Stock Asset Screen

**Stocks Detail Screen**

* Figure3.5 shows detail screen for particular stock which shows,
  + Candle chart with informative details on particular time for the same stock for evaluation of asset.
  + Line chart comparatively easy to understand for checking jitters in stock prices.
* Stocks also have a ticker to buy or sell the available stocks. Process to buy and sell is very easy, where the user has to select options by which they want to proceed with investments.
* Screen also has basic information about the asset like the founder, when this fund or stock has started, it’s important static values like P/E, P/B ratios etc.
* To verify the success or failure in the system, we provide toast as shown in Figures 3.6.



When a user performs operations, these pop ups will verify that process is success or failure as a notification.

Figure 3.6-Notification Toasts

**Watchlist**

* Figure3.7 shows watchlist screen on system. Which helps to get access to the favorite assets quickly.
* If a user wants to look over their favorite stocks or mutual funds then they can add it to the watchlist for rapid access.

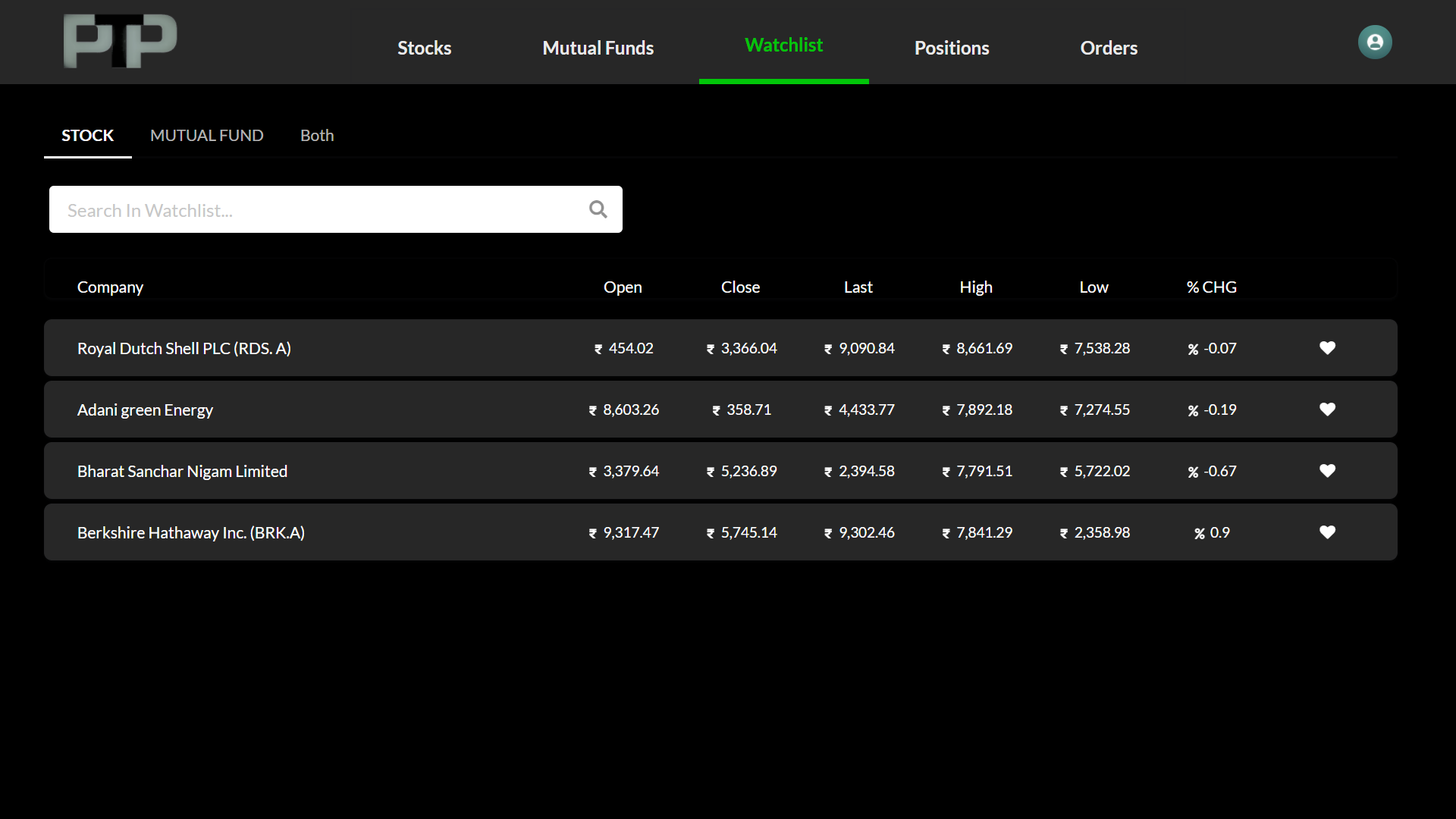
******

Figure3.7.1-Watchlist Stocks

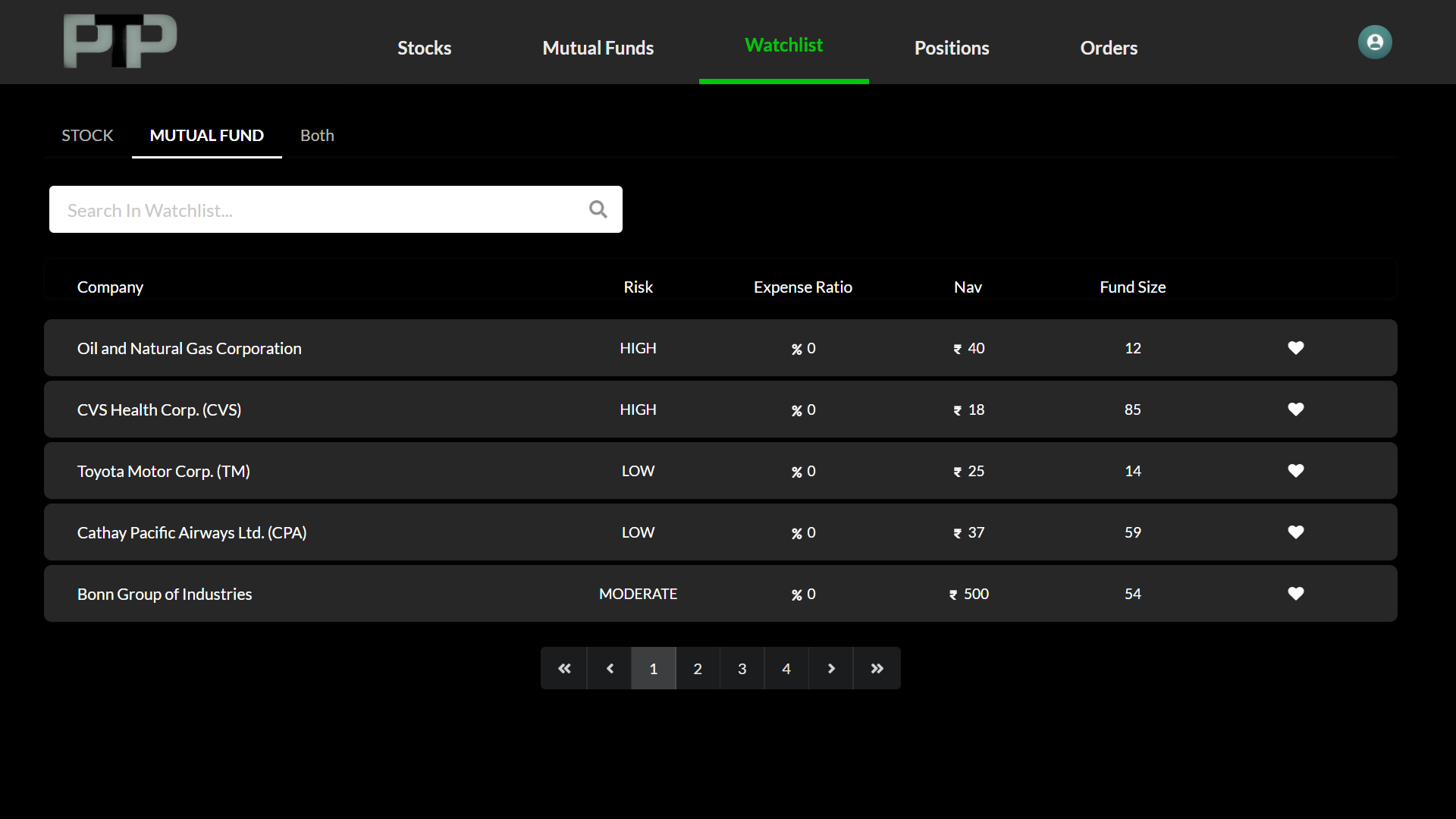
**

Figure3.7.2-Watchlist Mutual Funds

**Orders**

* Users also want to track their investment and see the live changes in prices as per the market change.
* Order screen help and provides all the necessary details regarding Bought assets, and also shows all the corresponding calculations regarding profit/loss percentages and position of their earning.

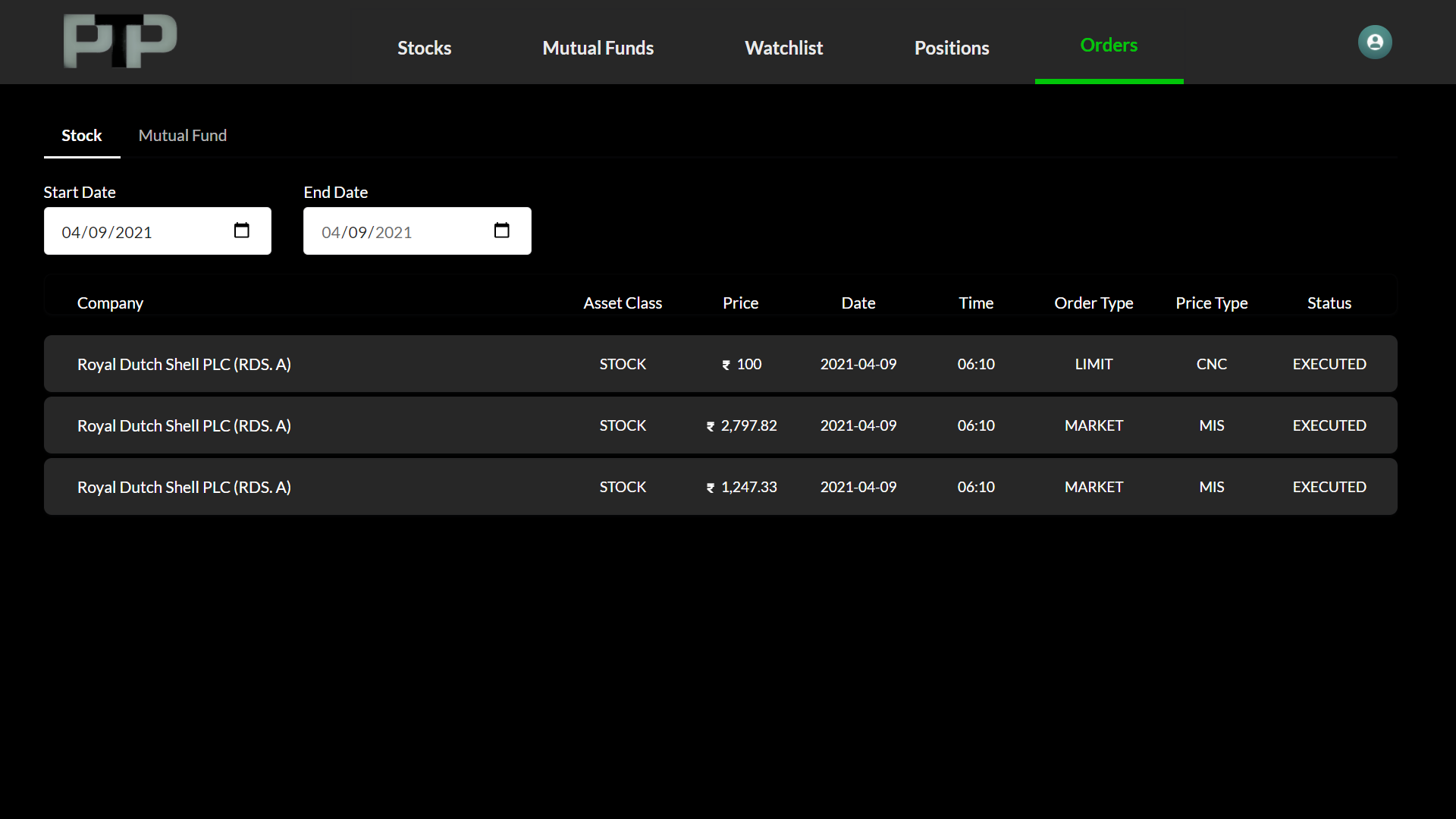
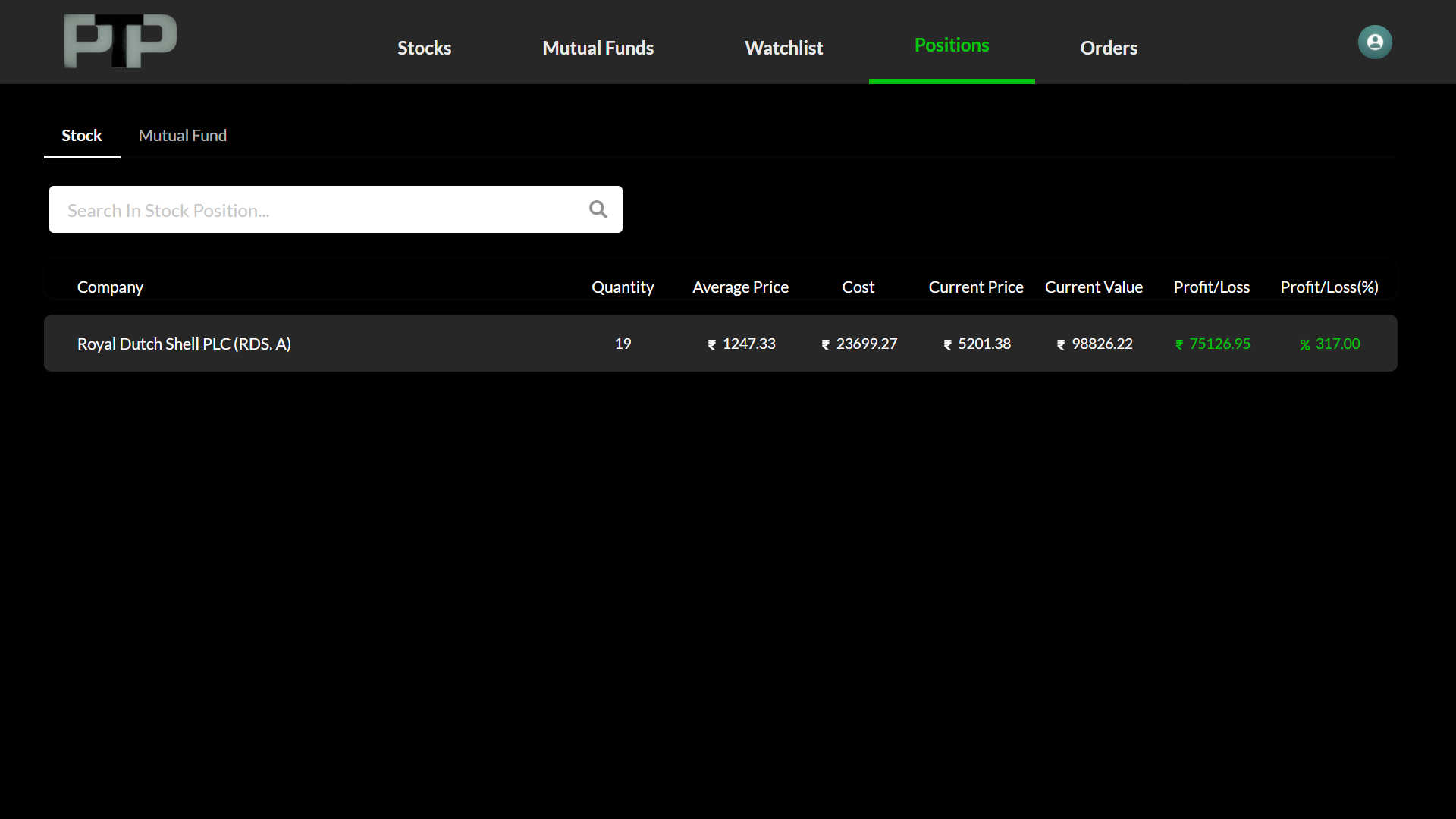
******

Figure3.8-Order Blotter

**Position**

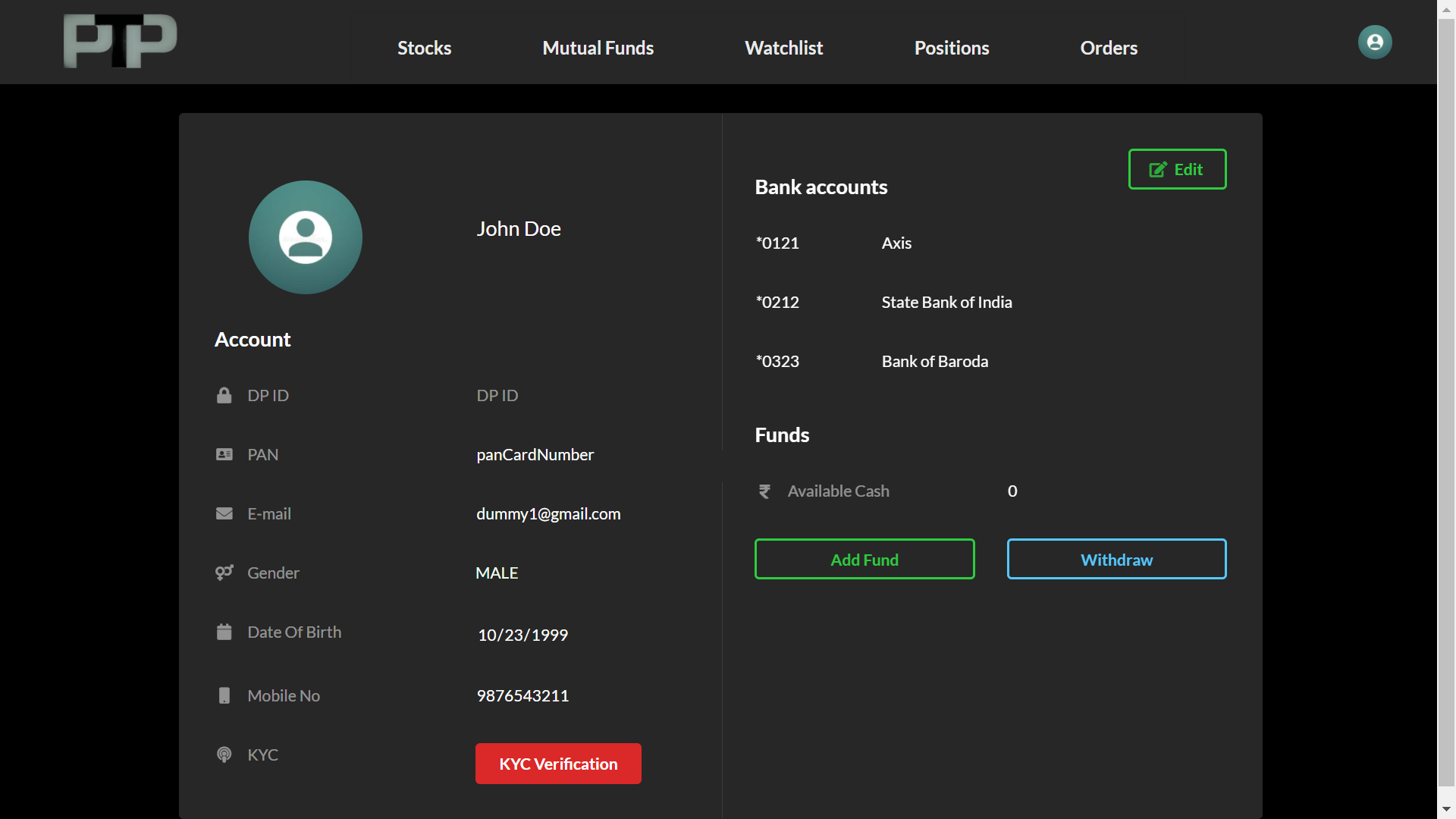
* Users can see their portfolios here. Their current standing for particular assets, they bought. Also, able to see the live price jitters in the market to decide whether to keep this stock or sell it.

******

*Figure3.9-Portfolio*

**Users Profile**

* Figure3.10, shows the user profile, they can update it directly from here using edit action on top right corner of the detail box.
* The KYC field shows whether the user has verified it or not. Users can request KYC verification using KYC Verification action.



*Figure3.10-User Profile*

**4.2 Database**

**Postgres**

* PostgreSQL is a powerful, open-source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.
* We will be using it as a data store. We have data in the form of tables. And Postgres is easy to maintain for the same.

As per **Pirimid Fintech LLP** Policies, the corresponding design and code cannot be shared.

**4.3 Deployment**

**GitLab - Docker - Kubernetes**

* Docker is a set of platforms as a service product that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels
* We are using it to create images and give it to production for fast development with Kubernetes.
* Gitlab will be used for CI/CD development and images will be pushed to Kubernetes containers for continuous development, testing and deployment.

**5**

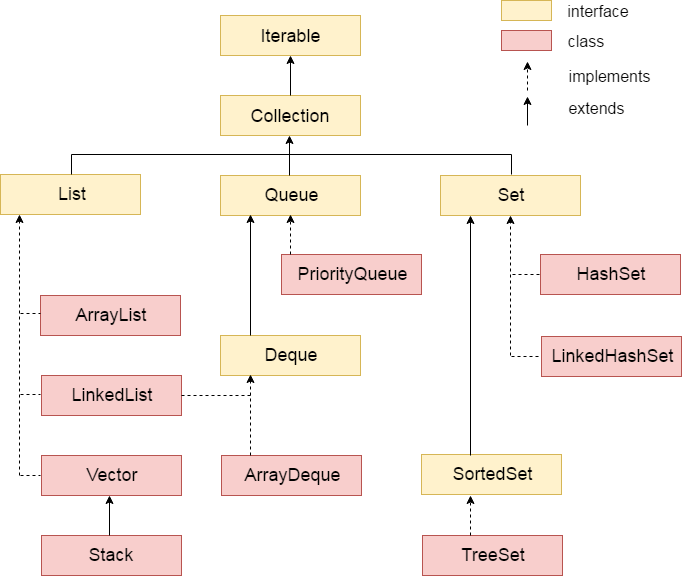
**Implementation**

**5.1 Java Spring Boot**

* Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.
* Spring framework provides flexibility to configure beans in multiple ways such as XML, Annotations, and JavaConfig. With the number of features increased the complexity also gets increased and configuring Spring applications becomes tedious and error-prone.
* **Spring is very popular for several reasons**
  + Spring’s dependency injection approach encourages writing testable code
  + Easy to use but powerful database transaction management capabilities
  + Spring simplifies integration with other Java frameworks like JPA/Hibernate ORM, Struts/JSF/etc. web frameworks
  + State of the art Web MVC framework for building web application

**5.2 Collections**

* Collections in java is a framework that provides an architecture to store and manipulate the group of objects. All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.
* Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (Array List, Vector, LinkedList, Priority Queue, HashSet, Linked HashSet, Tree Set etc.).



**Fig.5.1 Java Collections**

**5.3 Lambda Expression**

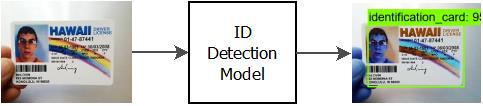
* The Lambda expression is used to provide the implementation of an interface which has a functional interface. It saves a lot of code. In the case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.
* Java lambda expression is treated as a function, so compiler does not create .class file.
* One issue with anonymous classes is that if the implementation of your anonymous class is very simple, such as an interface that contains only one method, then the syntax of anonymous classes may seem unwieldy and unclear. In these cases, you're usually trying to pass functionality as an argument to another method, such as what action should be taken when someone clicks a button. Lambda expressions enable you to do this, to treat functionality as method argument, or code as data.
* A lambda expression represents an anonymous function. It comprises of a set of parameters, a lambda operator (->) and a function body.

**5.4 Python Models**

Models Used

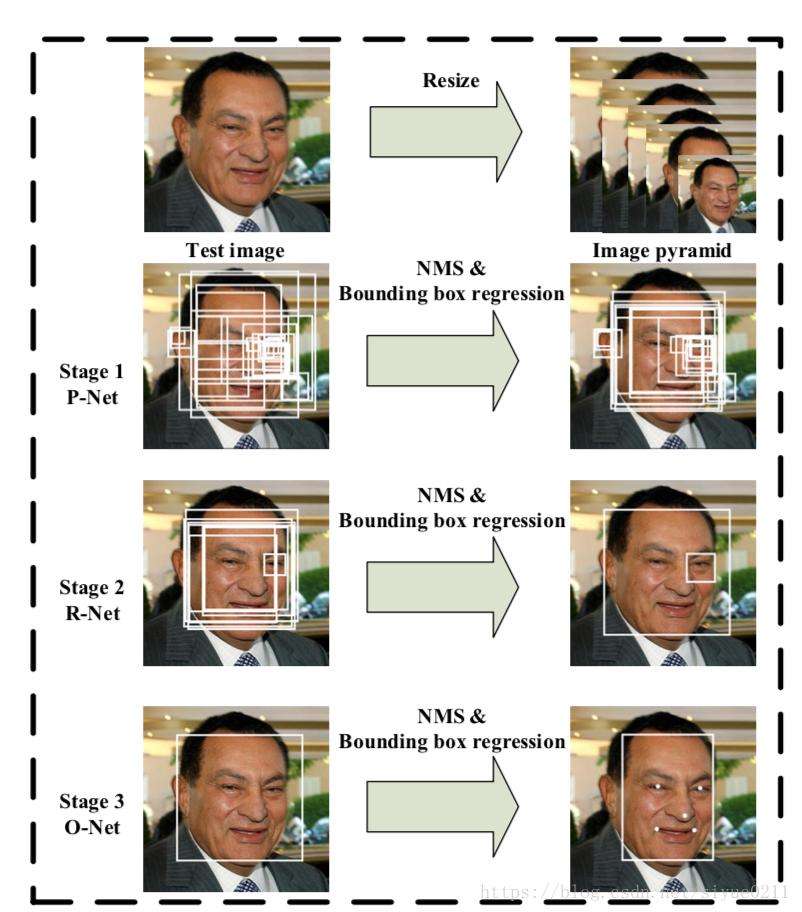
1. ID Detection
2. MTCNN
3. Face Net

* ID Detection
  + Specially trained model to detect ID card in an Image and return cropped image of just the ID card if detected.
  + It also gives confidence score of image being an ID card



**Fig.5.2 ID Card Detection Model**

* MTCNN
  + MTCNN works in 3 stages.
  + Stage 1 is called P-net (Proposal Net). Image is passed through shallow CNN. This will detect lot of faces in an image and also give a confidence score. It will make boundary boxes around face it detected.
  + Stage 2 is called R-net (Refine Net). This network will filter out a large number of boxes with poor effect, and finally select the candidate frames. And coincident boxes are removed using Non-Maximum Suppression (NMS) method.
  + Stage 3 is called O-net (Output Net). This network is similar to R-net. The difference is that this layer structure will identify the area of ​​the face through more supervision. It will pin point the location of facial feature like eyes, mouth and nose.



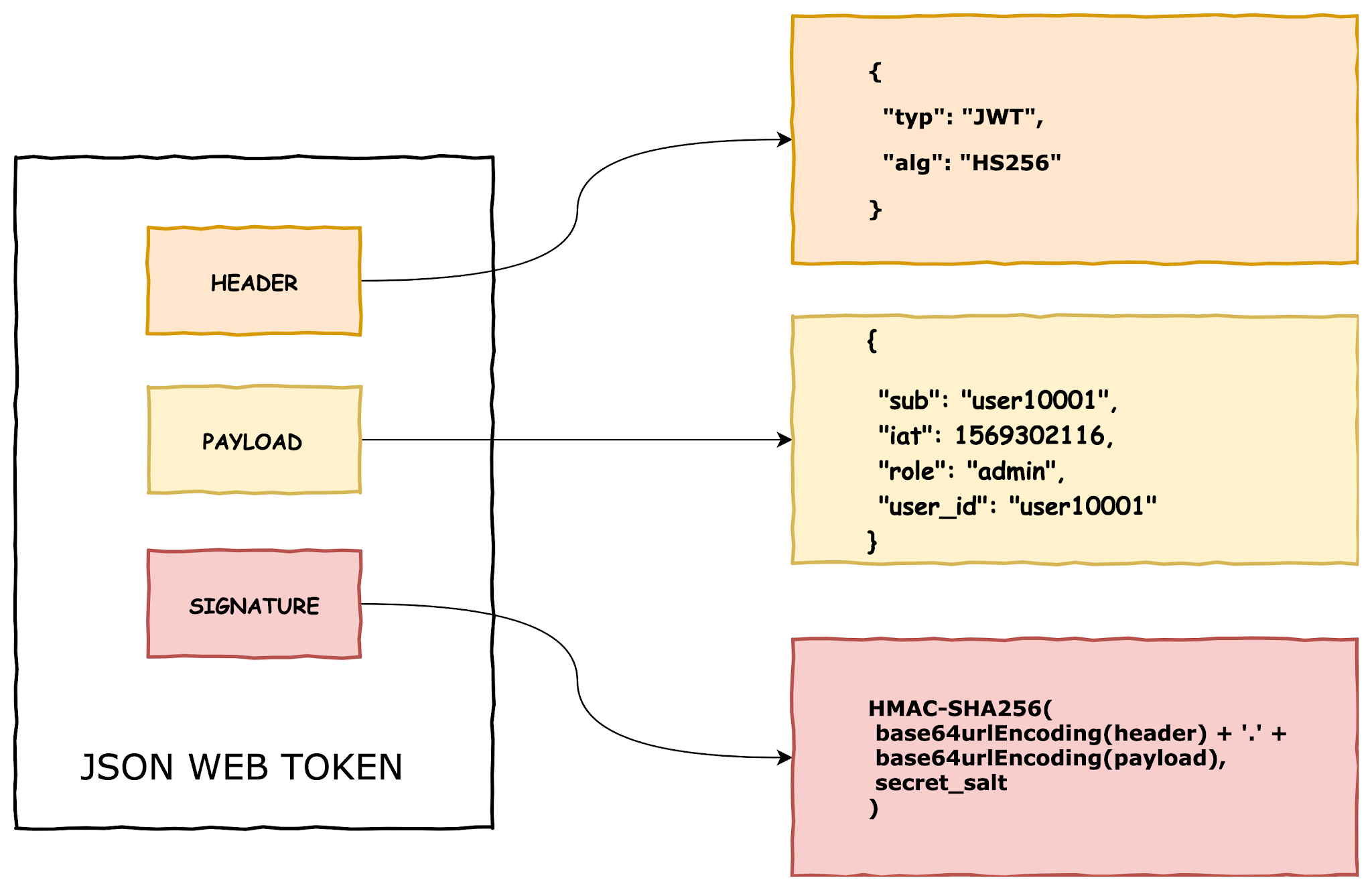
**Fig.5.3 MTCNN Stages**

* Face Net
  + Face Net uses deep convolutional networks along with triplet loss to achieve state of the art accuracy.
  + It maps each face image into a Euclidean space such that the distances in that space correspond to face similarity, i.e., an image of person A will be placed closer to all the other images of person A as compared to images of any other person present in the dataset.
  + Once we have embedding, we can perform different tasks like classification, verification or clustering.

**5.5 JSON Web Token**

For authentication we have used JWT token. JSON Web Token (JWT) is an open standard ([RFC 7519](https://tools.ietf.org/html/rfc7519)) for securely transmitting information between parties as a JSON object.

It is compact, readable and digitally signed using a private key/ or a public key pair by the Identity Provider (IdP). So, the integrity and authenticity of the token can be verified by other parties involved.



**Fig.5.5 JWT Structure**

**5.6 Semantic UI**

Semantic UI is a front-end development framework similar to bootstrap designed for theming. It contains pre-built semantic components that help create beautiful and responsive layouts using human-friendly HTML.

**5.7 SockJs**

For getting pricing data for Stock, we are using WebSocket. Client Subscribe to a particular Organization and receive data from WebSocket for that we are using sockjs client.

**6**

**Testing**

**6.1 Testing Plan**

The testing is a technique that is going to be used in the project is black box testing the expected inputs to the system are applied and only the outputs are checked.

**6.2 Testing Strategy**

The development process repeats this testing sub process a number of lines for the following phases.

* Unit Testing
* Integration Testing

Unit Testing tests a unit of code after coding of that unit is completed. Integration Testing tests whether the previous programs that make up a system, interface with each other as desired. System testing ensures that the system meets its stated design specifications. Acceptance testing is testing by users to ascertain whether the system developed is a correct implementation of the software requirements specification. Testing is carried out in such a hierarchical manner that each component is correct and the assembly/combination of components is correct. Merely testing a whole system at the end would most likely throw up errors in components that would be very costly to trace and fix. We have performed both Unit Testing and System Testing to detect and fix errors.

**6.3 Testing Methods**

We have performed Black-box testing for the testing purpose. A brief description is given below:

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing unit, integration, system and acceptance. It typically comprises most if not all higher-level testing, but can also dominate unit testing as well.

**6.4 Library & Frameworks**

* Frontend

We have used Chai and Mocha for testing react components.

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any JavaScript testing framework.

Mocha is a feature-rich JavaScript test framework running on Node. js and in the browser, making asynchronous testing.

* Backend

We have used Mockito and Junit for testing Java controllers.

Junit is an open-source unit testing framework for Java. Mockito is used when class under test should not be Mocked. Let's assume you are connecting to a remote service which is being built by one of your engineering team, and you are not familiar with its internal functionality but you know what requests and responses it returns. In that case, you can create a Mock of that Object, and define it with a set of responses returned in different situations. Each situation should get its own different test and for each response you should check separately the reaction of the code.

**7**

**Conclusion and Future Extension**

**7.1 Conclusion**

All of the modules in this project developed by us using React+SpringBoot frameworks. For the front-end design, we used React.js along with SemanticUI to implement all views. We are passing data in JSON format to the backend and parsing it using JACKSON. All database operations are done using JPA. All modules were developed separately and were then integrated.

The main concept behind creating communication between two systems lies in the usage of API of the respective system and the stability of that API. These features were developed by keeping in mind all the modules of the system. Hence the quality of features desired by the user is received.

**7.2 Future Extension**

* More User attractive UI.
* Live video recording in KYC instead of image.
* Host application on AWS.
* User can create multiple Watchlists instead of single per Assets.
* Users can perform Short Sells.
* Connect to third party API for pricing.
* Integration with Bank API.

**8**

**Bibliography**

Java documentation - <http://docs.oracle.com/en/>

Stack Overflow - <http://stackoverflow.com/>

Collection - <https://www.javatpoint.com/collections-in-java>

Hibernate - <https://www.javatpoint.com/hibernate-tutorial>

Spring Boot - <https://www.baeldung.com/spring-boot>

React - <https://reactjs.org/tutorial/tutorial.html>

Nextjs - <https://nextjs.org/>

Semantic UI - <https://react.semantic-ui.com/>