TRADING APPLICATION

PROBLEM STATEMENT

- **Design a web UI frontend** - facilitates making trade, viewing history, viewing status and delete or edit them by making requests to the REST API.

- **TradeDB** - a mongoDB to for proper retrieval and storage of data.

- **Trade REST API** - should be written in Java as a spring boot REST application.

- **Dummy Trade Fulfilment Service** - reads Trade records from the "TradeDB" and simulate fulfilling or rejecting those Trades and set the status like PROCESSING, FILLED, REJECTED.

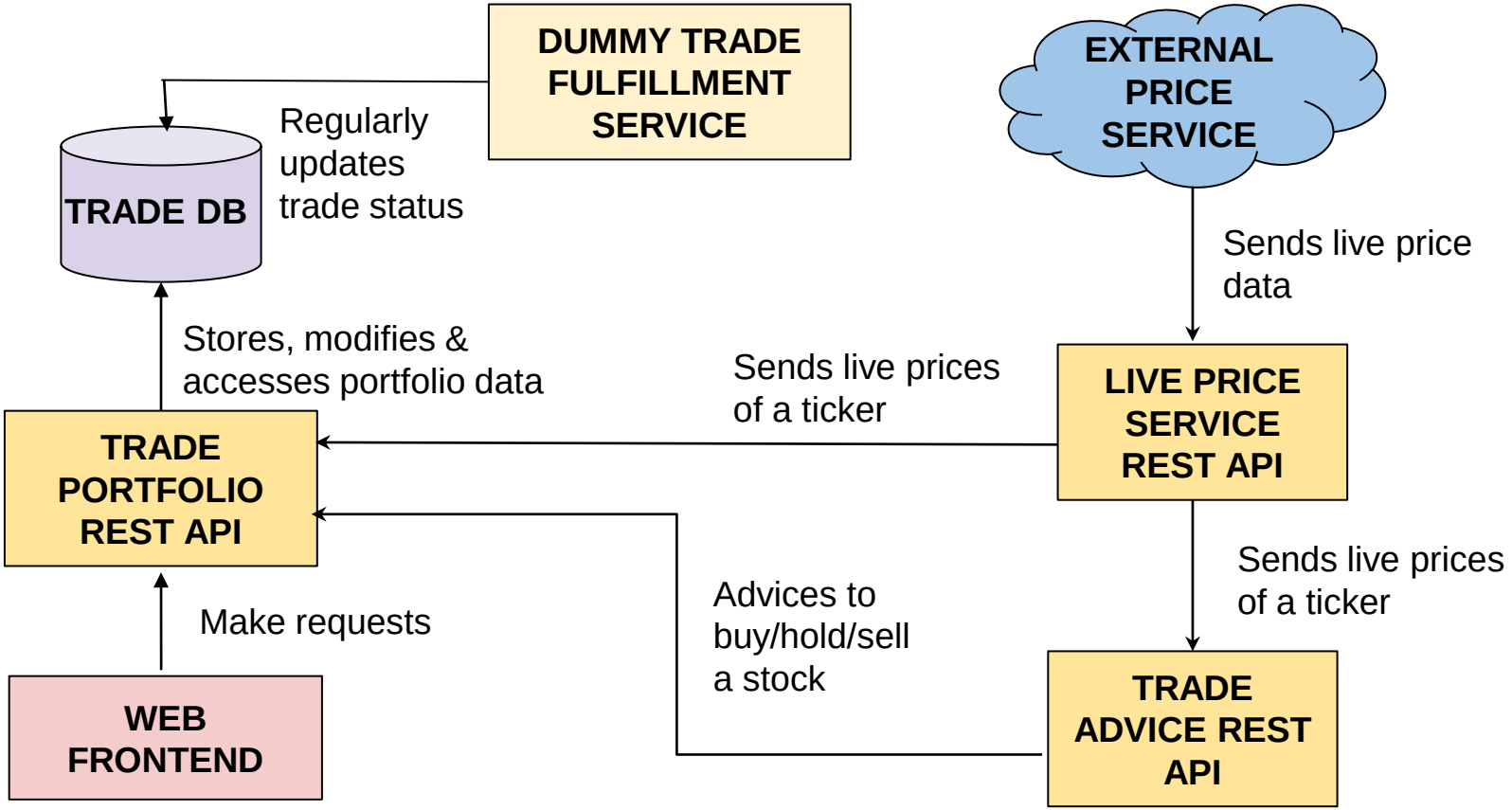
PROBLEM STATEMENT - EXTENSIONS

- **Portfolio REST API** - keeps track of a personal portfolio investments
- **Live Price Service REST API** - makes price data available to your other components through a REST API.
- **External Price Service** - This is an External service e.g. Yahoo Finance.
- **Trade Advice REST API** - returns an answer indicating whether the advice is to BUY, SELL or HOLD for that specific stock by analysing the previous stock prices.

TECHNOLOGY STACK

- **WIREFRAME** - LUCIDCHART
- **UI** - jQUERY, HTML, CSS
- **BACKEND** - SPRING BOOT JAVA
- **DATABASE** - MONGODB ATLAS
- **EXTERNAL PRICE SERVICE** - YAHOO FINANCE API
- **TESTING** - SPRING BOOT JUNIT TESTING
- **DEPLOYMENT** - DOCKER,JENKINS
- **SOURCE AND VERSION CONTROL** - BITBUCKET
- **TASK MANAGEMENT** - TRELLO, GOOGLE MEET

ARCHITECTURE DIAGRAM



TRADE PORTFOLIO API

- Trade Portfolio API is used to keep track of personal portfolio investments.

Authentication:

- A new user is allowed to sign up into the application.
- An existing user is allowed to log in to the application using his/her username and password.

Portfolio API operations for every user:

- Add some stocks for a particular ticker to his/her portfolio in the database.
- View details and status of a particular trade.
- Modify the quantity of stocks for a completed trade if it has not been processed yet.
- Delete a trade from view history

TRADE PORTFOLIO API

Dummy Trade Fulfillment Service:

- Regularly reads trades from the TradeDB and updates status of the trades in the following manner:
  - CREATED -> PROCESSING
  - PROCESSING -> FILLED/REJECTED
- Slightly modified Dummy Trade service in order to suit the needs of our application.

CODE SNIPPET

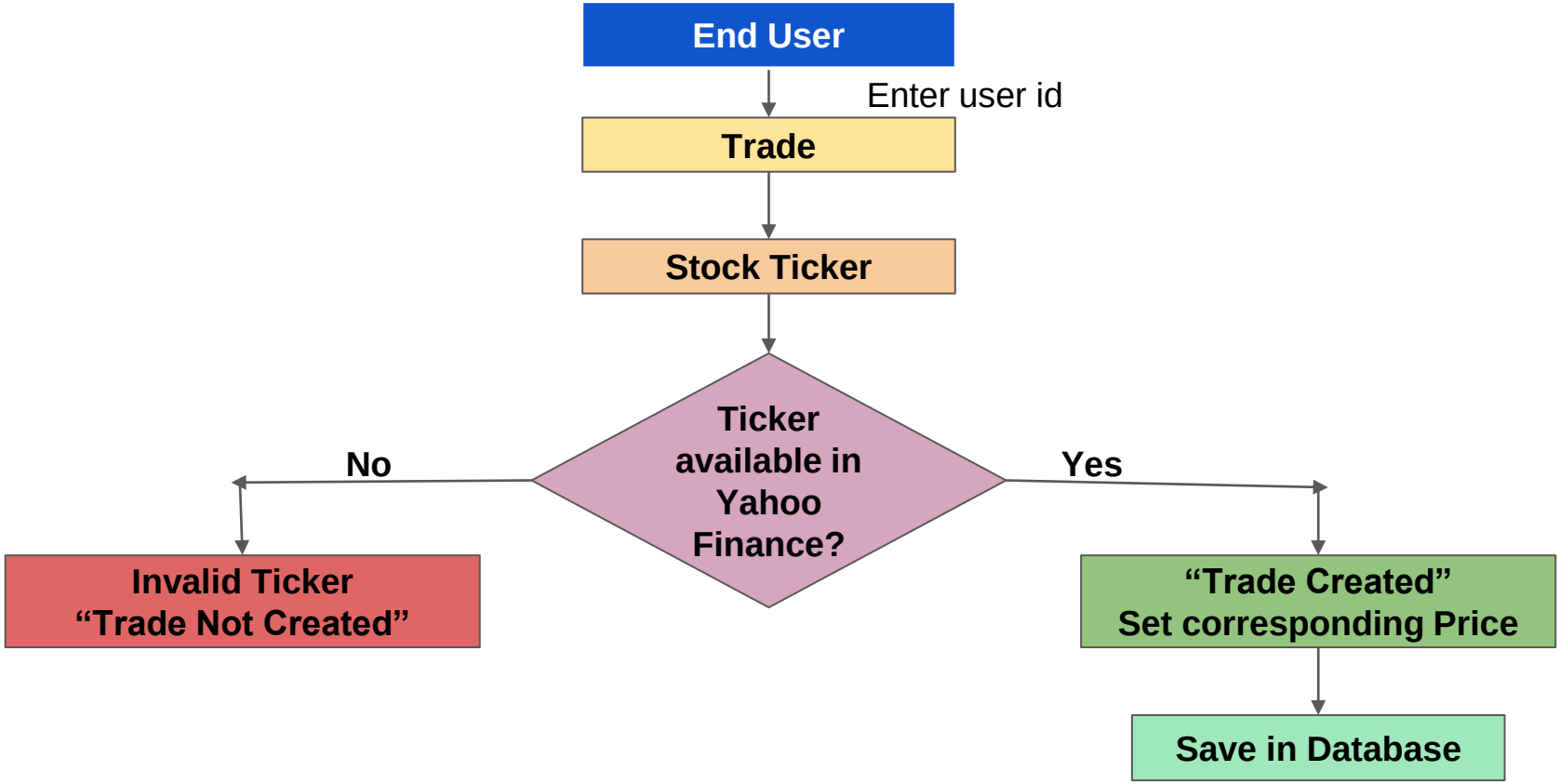
```java
public class Portfolio_Controller {
    @Autowired
    private User_Repository repository;


    @RequestMapping(value = "/signup", method = RequestMethod.POST)
    public void signup(@Valid @RequestBody User user) {
        repository.save(user);
    }

    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public User login(@Valid @RequestBody User user) {
        List<User> users = getUsers();
        for(User u: users)
        {

            if(user.getUsername().equals(u.getUsername()))
            {
                if(user.getPassword().equals(u.getPassword()))
                {
                    return getUserById(new ObjectId(u.get_id()));
                }
                else
                {
                    System.out.println("Password incorrect");
                }
            }
            else
            {
                System.out.println("User doesn't exist");
            }
        }
        return new User("","");
    }
```

```java
@RequestMapping(value = "/{id}/trades/{tradeid}", method = RequestMethod.GET)
public Trade getTradeById(@PathVariable("id") ObjectId id, @PathVariable ObjectId tradeid) {
  System.out.println("in gettradebyid");
  User user = repository.findBy_id(id);
  List<Trade> trades = user.getTrades();
  for(Trade trade: trades)
  {
      if(trade.get_id().equals(tradeid.toString()))
      {
          System.out.println("matching");
          return trade;
      }
      else
      {
          System.out.println("not matching "+trade.get_id()+" "+tradeid.toString());
      }
  }
  return new Trade();
}
```

LIVE SERVICE

- This service focuses on directly fetching real time live data from Yahoo Finance API.
- Trade tickers are taken from the selected trades.
- Availability of ticker in Yahoo Finance API is checked and the corresponding price is taken.
- Trade is created successfully and updated in Trade Portfolio Rest API and the database.

FLOW DIAGRAM



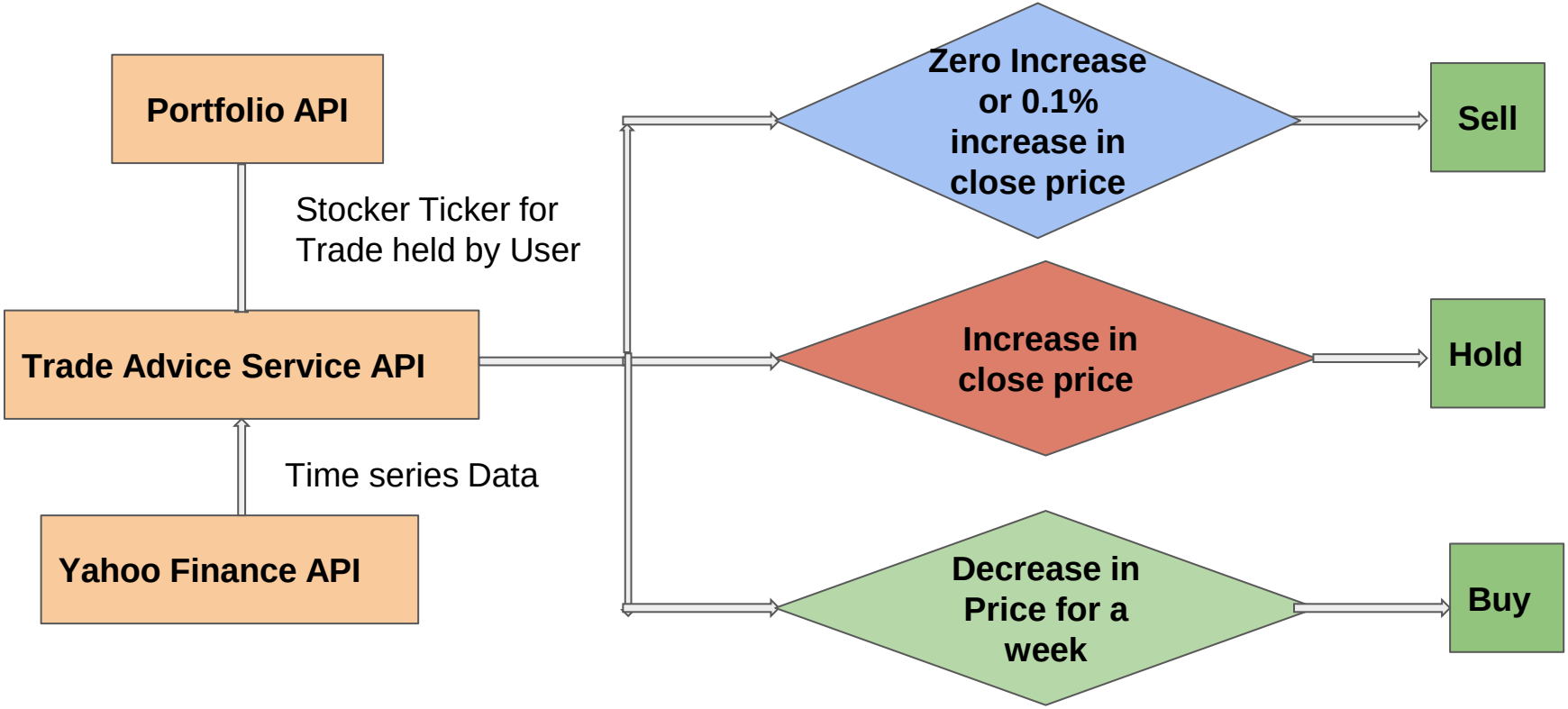
CODE SNIPPET

```java
@RequestMapping(value = "/{id}/trades", method = RequestMethod.POST)
public ResponseEntity<Object> createTrade(@PathVariable("id") ObjectId id, @Valid @RequestBody Trade trade )
        throws IOException{
    //System.out.println(trade.getTicker());
    trade.set_id(ObjectId.get());
    trade.setCreated(new Date());
    trade.setState(TradeState.State.CREATED);
    String[] symbols = new String[] {"INTC", "BABA", "TSLA", "AIR.PA", "HSBC", "JPM", "WFC", "BAC", "SAN", "RC", "GS", "LYG", "USB", "UBS", "MS", "AXP",
    double price [] = new double[symbols.length];
    int flag = 0;
    for (int i=0; i<symbols.length; i++) {
       price[i] = YahooFinance.get(symbols[i]).getQuote().getPrice().doubleValue();
        }
    for(int j=0;j<symbols.length; j++) {
        if(trade.getTicker().equals(symbols[j])) {
            trade.setPrice(price[j]);
            flag = 1;
            User user = repository.findBy_id(id);
            List<Trade> trades = user.getTrades();
            if(trades == null)
                trades = new ArrayList<Trade>();
            trades.add(trade);
            user.setTrades(trades);
            repository.save(user);
            break;
        }
        else {
            flag = 0;
        }
    }
    if (flag == 0) {
        System.out.println("Invalid Ticker");
        return new ResponseEntity<>("Invalid Ticker!! Trade not created", HttpStatus.FORBIDDEN);
    }
```

TRADE ADVICE SERVICE API

- Pull time series data (1 week) Data from Yahoo Finance API.
- List of historical quotes are obtained using getHistory() function.
- Close data of each day is used for providing advice to (SELL, BUY or HOLD).
- Simple Thresholding algorithm is used for providing the advice.

THRESHOLDING ALGORITHM

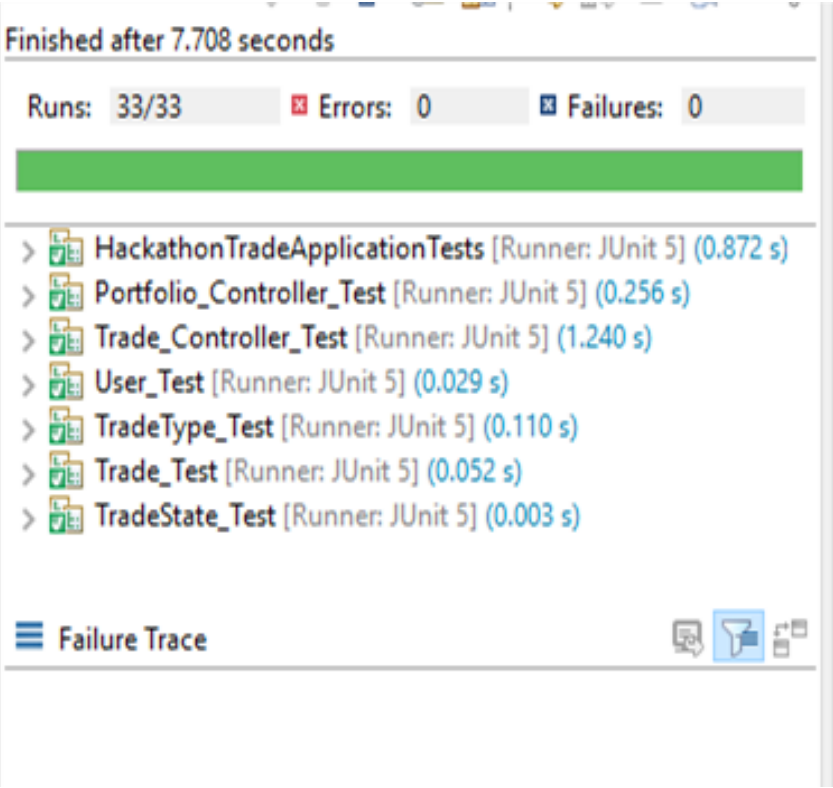


CODE SNIPPET

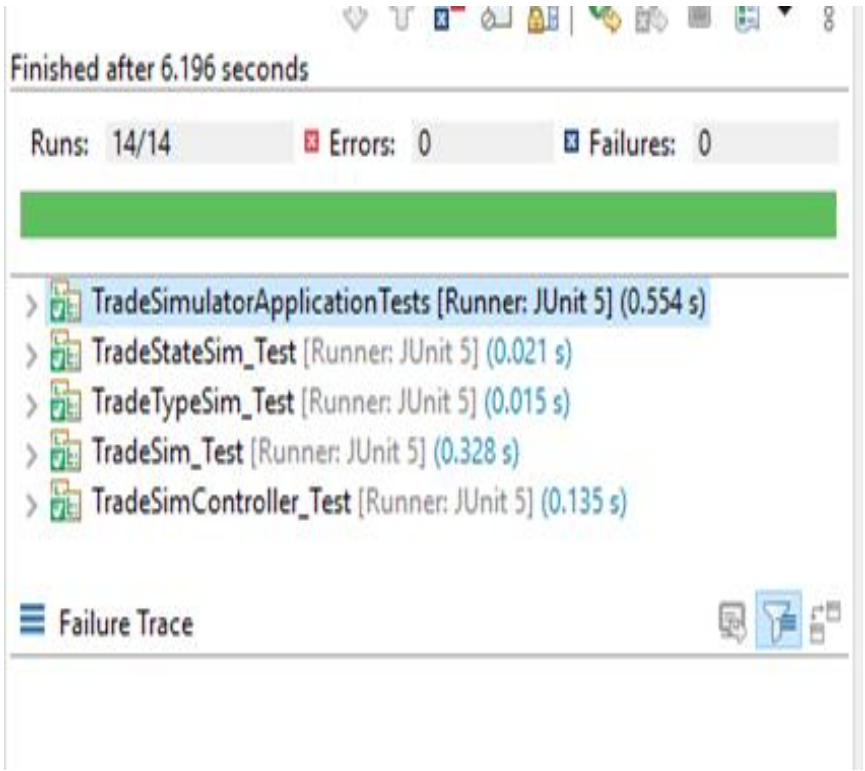
```
for(Trade t: trades)
  {
      if(t.get_id().equals(tradeid.toString()))
      {
          String tradeticker = t.getTicker();
            Calendar from = Calendar.getInstance();
            Calendar to = Calendar.getInstance();
            from.add(Calendar.WEEK_OF_MONTH, -1);

            Stock ticker = YahooFinance.get(tradeticker);
            List<HistoricalQuote> tickerHistQuotes = ticker.getHistory(from, to, Interval.DAILY);
            double previous_value = tickerHistQuotes.get(0).getClose().doubleValue();
            double current_value = tickerHistQuotes.get(tickerHistQuotes.size()-1).getClose().doubleValue();
            double current_value_1 = tickerHistQuotes.get(tickerHistQuotes.size()-2).getClose().doubleValue();

            double sum = 0.0;
            for(int j=0;j<tickerHistQuotes.size(); j++) {
                sum = sum+tickerHistQuotes.get(j).getClose().doubleValue();
            }
            sum = sum/tickerHistQuotes.size();

            double no0_1 = current_value * 0.001;
            double current_value0_1_high = current_value + no0_1;
            double current_value0_1_less = current_value - no0_1;

            if(sum >= current_value0_1_less && sum <=current_value0_1_high) {
                return "SELL";
            }
            else if((current_value < current_value_1) && (sum < previous_value)) {
                return "BUY";
            }
            else  {
                return "HOLD";
            }
```

SPRING BOOT JUNIT TESTING

Trade Portfolio Rest API



Dummy trade fulfillment service



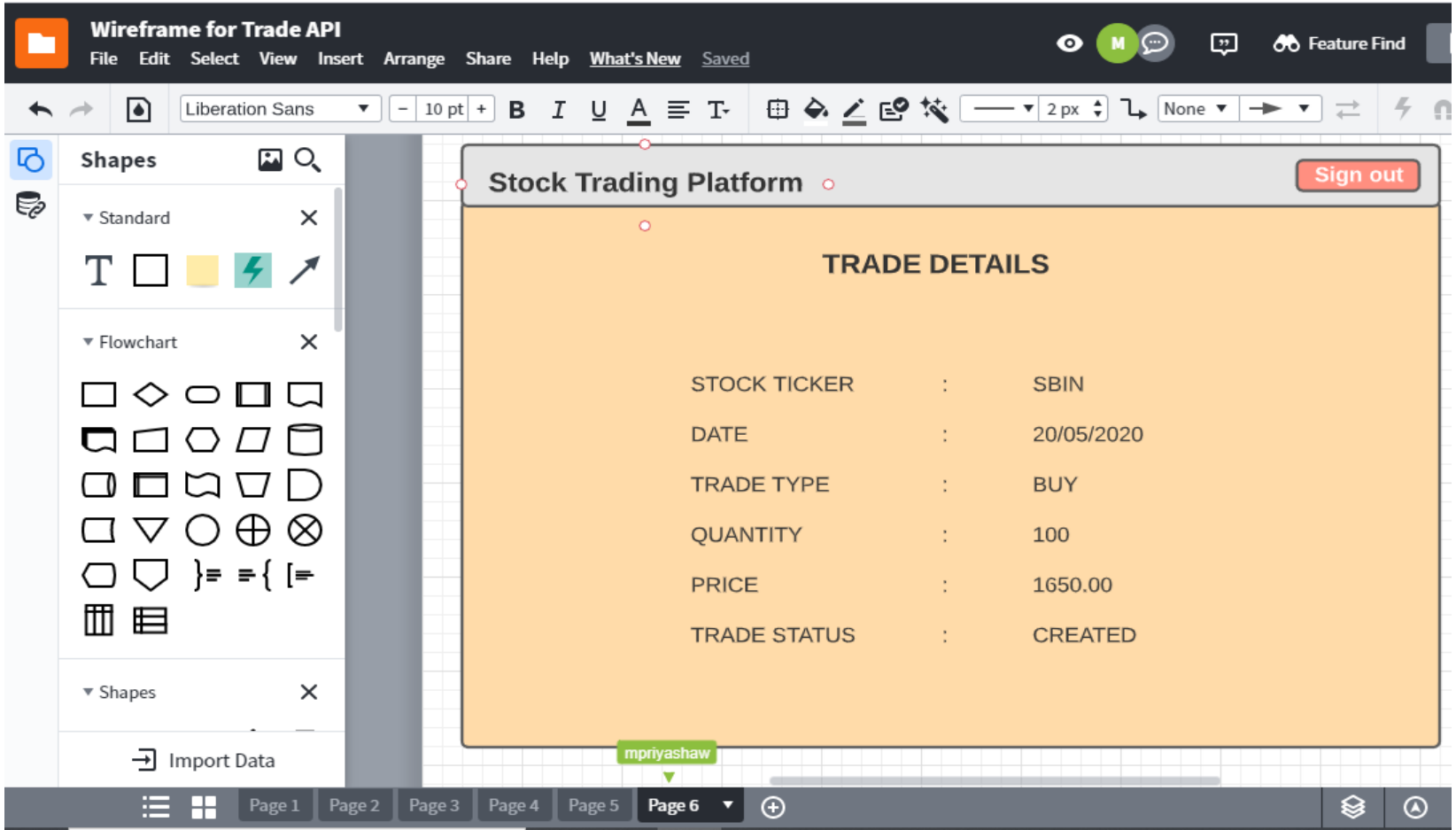
CODE SNIPPET

```java
@Test
void testCreateTradeStatusForbidden() throws Exception {
    Trade u = new Trade(new ObjectId(), new Date(), TradeState.State.CREATED, TradeType.Type.BUY, "SBIN", 10, 44.5);
    ResponseEntity<Object> entity = userController.createTrade(u);
    HttpStatus statusCode = entity.getStatusCode();
    assertEquals(HttpStatus.FORBIDDEN, statusCode);
}

@Test
void testCreateTradeStatusOk() throws Exception {
    Trade u = new Trade(new ObjectId(), new Date(), TradeState.State.CREATED, TradeType.Type.BUY, "HSBC", 10, 44.5);
    ResponseEntity<Object> entity = userController.createTrade(u);
    HttpStatus statusCode = entity.getStatusCode();
    assertEquals(HttpStatus.OK, statusCode);
}

@Test
void testUpdateStatusokl() {
    Trade u = new Trade();
    u.set_id(new ObjectId("5f8022c519173f2f09dacf00"));
    String status = "PROCESSING";
    when(userRepository.findBy_id(new ObjectId("5f8022c519173f2f09dacf00"))).thenReturn(u);
    u.setState(TradeState.State.CREATED);
    if (u.getState().equals(TradeState.State.CREATED)) {
        ResponseEntity<Object> entity = userController.updateStatus(new ObjectId("5f8022c519173f2f09dacf00"),
                status);
        HttpStatus statusCode = entity.getStatusCode();
        assertEquals(HttpStatus.OK, statusCode);
    }
}
```

jQUERY CODE SNIPPET

```
function loginnow()
{
    let user = { "username":$("#username").val(),"password":$("#password").val() };
    $.ajax({
        url: '/users/login/',
        type: 'post',
        contentType: 'application/json',
        data: JSON.stringify(user),
        datatype: 'json',
        success: function(response){
            $("#username").text("");
            $("#password").text("");
            $('#login').prop('disabled', true);
            localStorage.setItem("userid",response._id);
            window.location.replace("homepage.html");
        },
        error: function(response) {
            alert("Username/password incorrect");
        }
    });
}
```

WIREFRAME - LUCIDCHART



SOURCE AND VERSION CONTROL - BITBUCKET

trade-world

- Source
- Commits
- Branches
- Pull requests
- Pipelines
- Deployments
- Jira issues
- Downloads
- Repository settings

Snigdha Viswanathan / Final Hackathon

trade-world

Clone

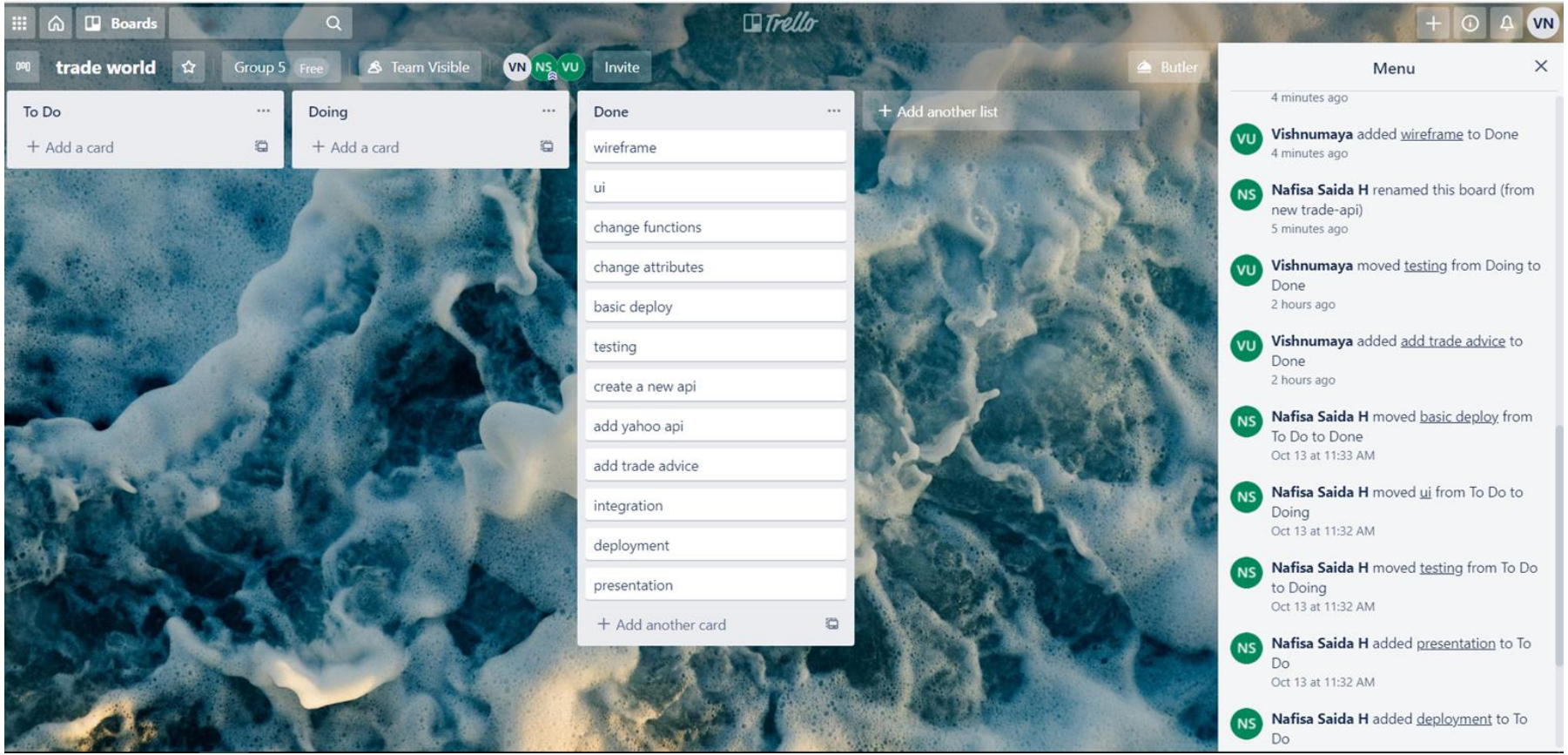
Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.

master | Files | Filter files

/

Name	Size	Last commit	Message
.mvn		6 days ago	Trade basic - Hackathon project
src		13 hours ago	Added colors to header, home, log out
.gitignore	411 B	19 hours ago	.gitignore edited online with Bitbucket
Dockerfile	135 B	6 days ago	Changed Docker file
mvnw	9.83 KB	6 days ago	Trade basic - Hackathon project
mvnw.cmd	6.45 KB	6 days ago	Trade basic - Hackathon project
pom.xml	1.83 KB	3 days ago	Added Live Service API and some HTML files

TRELLO TASK MANAGEMENT SYSTEM



FUTURE ENHANCEMENTS

- User authentication using spring security.
- Analysing the user’s history, alerts can be sent regarding any drastic changes in the stock prices
- Predictive Analytics tools integration

THANK YOU