Ftrade

Scenario 1:

The Ftrade app shows users a portfolio performance measure called Money Weighted Rate of Return (MWRR) that is delivered to the client via an API. Essentially, the client provides the ID of the user in an API call and the API returns the performance of that user's portfolio for multiple time periods. For example, for a particular user the API may return a -2% rate of return for the last week, +5% over the last month & +30% over the last year. For this system, please give an example of a unit test, an integration test and an acceptance test?

Acceptance test = Verify that the user can in the UI select the different period and values are updated and right.

Integration test =

1.1 Verify the API with existing portfolio type (1 week) and periods (1w, 1m, lastYear);

1.2 Verify the API with existing portofolio type (30 days) and periods (1w, 1m, lastYear);

1.3 Verify the API with existing portfolio type (360 days) and peridos (1w, 1m, lastYear);

1.3 Verify the API with wrong period time

1.4 Verify the API portfolio does not exist

Unit test =

1.2 Verify that method stop if period is null or empty

1.2 Verify that method stop if portfolio does exit or has not returns in db

1.3 Verify that method returns an Object like {"portfolio": "abc", "period": "1w", "mwrr":<value>}

Other option: the API could also return {"portfolio": "abc", "performance": [{"date":"2021-11-18", "mwrr":5}, {"date":"2021-11-19", "mwrr":-3}, {"date":"2021-11-20", "mwrr":<value>}]} . For instance, It is 7 days the array will have 7 objects.

Controller	**GET - hostname/{portfolioId}/MWRR?period= (1w, 1m, lastYear) response = {"portfolio": "", "period": "", "mwrr":""}**
Service Layer	

```
public Map<String, String> calculateMWRRByPeriod(String portfolioId, String period){
    HashMap<String, String> map = new HashMap<>();
    String from;
    if(period == null || period.length() == 0){
        return Map.of(); // throw error instead
    }
    switch(period)
    {
        case "1w":
            from = LocalDate.now().minusDays(7).toString();
            break;
        case "1m":
            from = LocalDate.now().minusDays(30).toString();
            break;
        case "lastYear":
            from = LocalDate.now().minusDays(365).toString();
            break;
        default:
            log.info("Period not supported");
    }

    List<MWRR> mwrrEntries = DAO.indexByPortfolioId(portfolioId, from);

    if(mwrrEntries.isEmpty()){
        return Map.of(); //throw error instead
    }

    long mwrrPeriod= CalculateMWRRAgregate(mwrrEntries);
    map.put("portfolio", portfolioId );
    map.put("period", period);
    map.put("mwrr", mwrrPeriod);
    return map;
}
```

Dao Layer

```
   public List<MWRR> indexByPortfolioId(String portfolioId) {
        String INDEX_BY_PORTFOLIO_ID = """
            select portfolioId, mwrr, data from WeightedRateReturn where portfolioId = :
portfolioId AND data >= :from
            """;
        Map<String, String> args = Map.of("portofolioId", portfolioId);
        return namedParameterJdbcTemplate.query(INDEX_BY_ACCOUNT_ID, args, (rs, rowNum) -> MWRR.
builder()
                .portfolio(rs.getString("portfolio"))
                .mwrr(rs.getString("mwrr"))
                .timestamp(rs.getString("date"))
                .build());
    }
```

MWRR will by a Type class

PortfolioId	MWRR	TimeStamp