Функциски барања:

1. Системот управува со извори на податоци. Системот мора автоматски да се поврзе со избран извор на крипто податоци како на пример Coingecko, Binance, Coinbase, Kraken.
2. Системот ја презема листата со врвните 1000 криптовалути-симболи по пазарна капитализација.
3. Системот ги идентификува невалидните или проблематичните валути. Тука спаѓаат валути кои се делистани, валути со премала ликвидност или нестабилна квота. Системот мора сите овие валути да ги филтрира без рачна интервенција.
4. За секоја криптовалута системот треба да провери кои податоци се веќе зачувани во базата.
5. За криптовалутите кои немаат зачувани податоци системот мора да превземе најмалку 10 години историски дневни “отвори високо ниско затвори волумен” (OHLCV) податоци или пак максимумот што API го поддржува.
6. Доколку податоците постојат потребно е системот да го земе последниот датум и да пресмета кои податоци недостасуваат, а потоа да ги додаде во базата/документот со филтрирани и финални податоци.
7. Системот мора да нуди механизам за преземање на сите податоци кои недостасуваат до тековниот ден.
8. Системот има Pipe-and-Filter трансформација.
9. Системот мора да имплементира филтер 1 (Extraction Filter) каде што презема листата од најдобрите-највредни 1000 валути, ги прочистува податоците и потоа прочистените податоци ги зачувува.
10. Системот мора да имплементира филтер 2 (Date Check Filter) каде што ја проверува базата и потоа враќа симбол заедно со последниот датум, за валутите кои имаат податоци за нив, а оние кои немаат податоци ги додава на листа.
11. Системот мора да имплементира филтер 3 кој што ги додава OHLCV податоците за сите симболи за кои нема записи и тоа додава податоци од 10 години наназад или максимално достапно време па до тековниот датум.
12. Системот мора да имплементира филтер 4(Data Completion Filter) каде што ги додава OHLCV податоците за сите симболи до тековниот датум.
13. Системот мора да ги зачува податоците во база или датотечен формат како на пример (CSV/JSON).
14. Системот мора да избегнува дупликати при додавање на нови редови.
15. Системот мора да биде конзистентност во форматите. Да има исти колони, исти датумски формат...со цел да се избегне настанувањето грешки.
16. Системот мора да прикаже грешки доколку API не е достапен

17. Системот мора да прави “retry” при неуспешен повик, односно да се обиде уште неколку пати.
18. Системот мора да ги прати/следи сите трансформации и грешки што ќе настанат.
19. Системот треба да измери колку време му треба да ја исполни празната база.
20. Системот треба да нуди интуитивен и лесен за користење кориснички интерфејс преку кој графички ќе се претставуваат податоците.
21. Системот треба да има домашна страна, на која ке нуди избор од симболи.
22. Системот треба да овозможи пребарување на симболи преку текстуално поле, пребарувањето мора да биде case-insensitive. Резултатите од пребарувањето треба да се прикажат во dropdown листа. Клик на резултат од dropdown треба да навигира до страницата со детали за избраниот симбол.
23. Системот треба да прикаже дневни податоци (open, high, low, close, volume, итн.) за избран симбол.
24. Системот треба да овозможи избор на датумски опсег (from/to) и филтрирање на податоците по овој опсег.
25. Системот треба да пресметува и прикажува: latest close, average volume, highest high за избраниот опсег.
26. Системот треба да прикажува процентуална промена на цената за избраниот период.
27. Системот треба да прикаже график со цени за дадената валута во избраниот опсег. Предефинирано ке биде приказ за последните месец дена.
28. На почетната страница системот треба да прикажува месечни агрегации за најмалку 6 пред-дефинирани симболи за последните 12 месеци.
29. На страницата за детали системот треба да прикажува дневни податоци за избраниот симбол во рамки на избраниот датумски опсег.
30. Системот треба табеларно да ги прикаже сите дневни записи за избраниот симбол.
31. Табелата треба да ги прикаже сите атрибути на ентитетот CryptoSymbol.
32. Табелата треба да овозможи сортирање по датум.
33. Системот треба да овозможи export на тековно прикажаните (филтрирани) податоци за симбол во CSV формат. Името на фајлот треба да го содржи симболот.
34. Системот треба да овозможи посебна страница за приказ на сите записи од базата.
35. Приказот треба да користи virtual scrolling за да не се рендерираат сите редови одеднаш.
36. Податоците мора да се вчитуваат од backend со pagination.
37. Backend системот мора да обезбеди следни REST endpoints:GET /api?page={page}&size={size} – листа на записи со pagination. GET /api/all – алтернативен endpoint за сите податоци. GET /api/symbol/{symbol} – податоци за симбол. GET /api/symbol/{symbol}?from={date}&to={date} – податоци за симбол во датумски опсег. GET /api/id/{id} – податок по ID. GET /api/search?query={query} –

пребарување на симболи. Сите endpoints треба да враќаат податоци во JSON формат

Нефункциски барања:

1. Перформанси
   - Системот треба да обработи крипто валута во разумно време (помалце од секунда).
   - Системот треба да подржува 100 истовремени корисници.
   - Backend мора да користи pagination и да не враќа повеќе од 10 000 записи по барање.
   - Frontend листите со голем број записи мора да користат virtual scrolling за да се намали потрошувачката на меморија и да се подобри време на рендерирање.
   - Пребарувањето на симболи мора да користи debounce од најмалку 300 ms пред API повик.
   - За прикажување на повеќе графикони паралелно, frontend треба да користи паралелни API повици.
2. Скалабилност
   - Системот треба да поддржи додавање на нови извори на податоци без промена на целата архитектура.
   - Системот треба да може да ги пренесе податоците од CSV во SQL без голема измена.
   - Архитектурата мора да поддржува поголеми скупови податоци преку pagination и virtual scrolling, без значителен пад во перформансите.
3. Конзистентност
   - Податоците мора да бидат во стандарден формат.
4. Безбедност
   - Пристапот до базата мора да се врши преку JPA/Hibernate со параметризирани queries за да се избегне SQL injection.
   - Внесените нумерички податоци мора да се валидираат (на пример со @PositiveOrZero).
   - CORS мора да биде конфигуриран така што ќе дозволува пристап од фронтенд доменот

5. Одржливост
   - Филтрите мора да бидат независни единици на код.
   - Промената на еден филтер не смее да влијае на другите.

- Додавањето на нови функционалности не смее да се одрази на веќе постоечките.

6. Квалитетен код
   - Кодот треба да ги следи принципите кои обезбедуваат читливост и разбирливост
   - Кодот треба да го следи SOLID и DRY принципот
   - Кодот треба да биде лесно одржлив
   - Кодот треба да биде робустен и ефикасен.

7. Употребливост
   - Интерфејсот потребно е да биде едноставен и лесен за користење за анализа на податоци.

Персони

Персона 1: „Casual Crypto Viewer“ (Ана, 23 години)
Профил: Студентка или млад професионалец, сака да следи неколку познати крипто симболи (BTC, ETH, SOL).
Цели: Брз визуелен преглед на трендови, последни цени и едноставни графици без длабока анализа.
Однесување: Го користи системот неколку пати неделно, најчесто ја посетува почетната страница и деталите за 1–2 симболи.

Персона 2: „Data-driven Enthusiast“ (Марко, 30 години)
Профил: IT или finance ентузијаст кој сака да анализира историја, волатилност и волумен по датуми.
Цели: Филтрирање по период, споредба на волумен и цени, како и експорт на податоци во CSV за понатамошна анализа (Excel, Python).
Однесување: Често го користи date range филтерот, CSV export, табеларен приказ и детална статистика за симбол.

Персона 3: „Guest / Demo User“ (Игор, 19 години)
Профил: Нов корисник кој првпат гледа крипто податоци и нема напредно знаење.
Цели: Лесно да се снајде, да разбере што значат графиконот, цената и промената и да може да „шета“ низ системот без да се изгуби.
Однесување: Главно користи ticker, grid и basic search, без сложени филтри.

Наратив за кориснички сценарија

1.Ана ја отвора почетната страница. Во полето за пребарување внесува „BTC“. По кратка пауза (debounce) системот праќа барање до /api/search?query=BTC и прикажува dropdown со симболи како „BTCUSDT“, „BTCUSDC“ итн. Ана кликнува на „BTCUSDT“ и системот ја пренасочува на страницата /symbol/BTCUSDT, каде гледа графикон и табела со дневни податоци.

2.Марко се наоѓа на страницата /symbol/ETHUSDT. Почетно се прикажуваат податоците за последните 30 дена. Тој избира from = 2024-01-01 и to = 2024-01-31 и притиска „Continue“. Frontend праќа GET /api/symbol/ETHUSDT?from=2024-01-01&to=2024-01-31. Backend враќа филтрирани записи, графиконот и табелата се освежуваат и Марко ја гледа промената на цена и волумен токму за јануари 2024.

3.Откако ќе ги филтрира податоците за BTCUSDT за одреден период, Марко кликнува на „Download CSV“. Frontend ги зема тековните табеларни податоци, креира CSV-содржина (headers + rows), создава Blob и темпорарен линк, а прелистувачот автоматски го симнува фајлот BTCUSDT-data.csv.