|  |  |  |
| --- | --- | --- |
| توضیح | تسک | مرحله |
| Users, Portfolios, NAV History | طراحی جداول دیتابیس | 1 |
| با داده‌های تستی | نوشتن توابع NAV | 2 |
| پاسخ ساده به /start و /nav | پیاده‌سازی بات تلگرام | 3 |
| خواندن و نوشتن اطلاعات کاربر | اتصال بات به دیتابیس | 4 |
| خرید/فروش، به‌روزرسانی NAV | اضافه کردن بیزینس لاجیک معاملات | 5 |
| آپدیت NAV روزانه | اضافه کردن Cron Job | 6 |
| سرور/کلاود، تنظیم محیط، تست نهایی | دیپلوی | 7 |
|  |  |  |

**طرح کلی (High-Level Architecture)**

۱. **نیازسنجی و طراحی بیزینس لاجیک**

* تعریف دقیق محاسبات NAV: فرمول‌ها، ورودی‌ها، خروجی‌ها.
* تعیین نقش‌ها: کاربر عادی، ادمین، سرمایه‌گذار خاص؟
* مشخص کردن نوع داده‌ها: پرتفوی، معاملات، NAV روزانه، گزارش‌ها.
* ۲. **طراحی معماری سیستم**

Telegram Bot (فرانت)

Backend (API/Logic)

Database (ذخیره داده و تاریخچه).

در صورت نیاز: Cron Jobs برای آپدیت NAV روزانه.

۳. **طراحی دیتابیس**

* جداول: Users، Portfolios، Transactions، NAV History، Logs.
* انتخاب نوع دیتابیس:
  + PostgreSQL/MySQL → اگر دیتا ساختاریافته‌ست.
  + Redis → برای کش و سرعت.

۴. **محاسبات NAV**

* تعریف دقیق محاسبه ارزش خالص دارایی:

NAV = (Total Assets - Total Liabilities) / Units Outstanding

۵. **توسعه بات تلگرام**

* استفاده از Telegram Bot API (کتابخانه‌هایی مثل python-telegram-bot، aiogram یا برای Node.js telegraf).
* منطق بات:
  + دریافت دستورات (e.g. /nav، /portfolio، /buy، /sell).
  + پاسخ‌دهی با داده‌های پردازش شده از بک‌اند.

۶. **اتصال به دیتابیس**

* ORM (مثل SQLAlchemy برای پایتون).
* ایجاد session امن، هندل کردن خطاهای دیتابیس، لاگ‌گیری.

۷. **دیپلوی**

* انتخاب زیرساخت:
* سرور (VPS مثل Hetzner, DigitalOcean).
* یا سرویس‌های serverless/container (مثل Heroku، Railway، AWS).
* ست کردن متغیرهای محیطی: TOKEN، DB\_URI.
* راه‌اندازی SSL/HTTPS (اگر API باز دارین).
* Cron job برای به‌روزرسانی NAV دوره‌ای.

۸. **سکیوریتی و مانیتورینگ**

* جلوگیری از SQL Injection.
* محدودیت ریت برای جلوگیری از اسپم.
  + - * + لاگینگ و مانیتورینگ (Sentry، Grafana، یا حتی فایل لاگ ساده).
  1. اضافه كردن لايه امنيت دوم (براي withdraw) مثل sms ، ايميل يا Google authenticator نظر من ايميل هست اگه بتونم روي سرور ايميل رو پياده سازي كنم
  2. مساله دوم بحث rate limit هست. یعنی محدودیت تعداد درخواست توی ربات. که از chathgpt پرسیدم گفت هم میتونی توی nginx هم توی فلسک و هم توی خود اسکریپت ربات طراحی کرد
  3. رمزنگاری روی sqlite رو از طریق sqlcipher
  4. جدول audit log برای ثبت لاگ ها

۹. **تست و Rollout**

* تست یوزکیس‌ها با کاربر تستی.
* تست لود (اگه کاربران زیاد هستن).
* Rollout

به امنیت در چند لایه نگاه کنی: **کاربران، بات، بک‌اند، دیتابیس، و سرور**

**🛡️ نکات امنیتی مهم که باید لحاظ بشه**

**۱. امنیت API تلگرام و توکن‌ها**

* توکن بات رو اصلا نباید توی کد پابلیک (مثل GitHub) قرار بدی.
* از **متغیر محیطی (Environment Variable)** برای نگه‌داری توکن استفاده کن.
* دستورات حساسی مثل /admin رو فقط به IDهای مشخص اجازه بده.

**۲. احراز هویت و سطح دسترسی**

* اگر بات قراره اطلاعات مالی خاص نشون بده، حتماً **احراز هویت کاربران** (حداقل با ID تلگرام) داشته باش.
* کاربران رو دسته‌بندی کن (مثلاً: کاربر عادی، سرمایه‌گذار VIP، ادمین).
* حتماً برای عملیات حساس (مثل برداشت یا تغییر داده) **تأیید دوباره** بگیر.

**۳. امنیت ارتباطات**

* اگر بات به بک‌اند وصل می‌شه، ارتباط باید از طریق **HTTPS** انجام بشه.
* API Keyها و Database Credentials رو در فایل امن یا Secret Manager نگه دار.

**۴. امنیت دیتابیس**

* هرگز مستقیم Query ننویس (SQL Injection خطرناکه). از ORM یا Prepared Statement استفاده کن.
* حتماً دسترسی دیتابیس محدود باشه (مثلاً کاربر دیتابیس فقط CRUD لازم رو داشته باشه، نه همه‌ی دسترسی‌ها).
* از **Backup و رمزگذاری** (Encryption) برای داده‌های حساس استفاده کن.

**۵. Rate Limiting و جلوگیری از اسپم**

* برای جلوگیری از سوءاستفاده، ریت لیمیت بزار (مثلاً هر کاربر در دقیقه فقط X درخواست).
* لاگ بگیر و آنالیز کن که بات به شکل مشکوک استفاده نشه.

**۶. مدیریت خطاها و لاگ‌ها**

* خطاها رو لاگ کن ولی هرگز اطلاعات حساس (مثل رمز، توکن، NAV حساس) رو داخل لاگ ننویس.
* مانیتورینگ فعال داشته باش (مثل Sentry، Grafana، Prometheus).

**۷. آپدیت و پچ امنیتی**

* کتابخانه‌ها، سیستم‌عامل و سرور رو همیشه به‌روز نگه دار.
* اگر از Docker استفاده می‌کنی، ایمیج‌ها رو از منبع معتبر و آپدیت‌شده بگیر.

**۸. مکانیزم‌های ضد حمله**

* محافظت در برابر **Brute Force** (مثلاً اگر کسی بارها با اکانت فیک تست کنه).
* محافظت در برابر **Man-in-the-Middle** با SSL.
* جداسازی محیط توسعه و تولید (Dev/Prod Separation).

💡 **توصیه عملی:**  
برای پروژه‌های مالی، حتماً یک **Code Review امنیتی** داشته باش (یا حتی تست نفوذ (Pen Test)) قبل از رفتن به محیط عملیاتی، چون کوچک‌ترین باگ ممکنه به ضرر مالی ختم بشه.