تمرین دوم

محمدعرفان هومانفر

بخش اول:

Business Requirement

۱)داشبورد مدیریت کاربران:

صاحب سیستم داشبوردی جهت مدیریت کاربرانی که از سیستم استفاده میکنند را داشته باشد تا توانایی اضافه کردن کاربر جدید و درصورت نیاز غیرفعال کردن کاربران فعلی توسط ادمین اصلی سیستم وجود داشته باشند.

- در سیستم ما وجود پنل ادمین این امکان را فراهم میکند که کاربرانی که از سیستم استفاده میکنند مدیریت شوند.

۲) تعمیر و نگهداری سیستم مقرون به صرفه را از طریق اتوماسیون فعال باشد تا با اجرا و نصب خودکار و رفع اتوماتیک هشدار ها سرعت عمل افزایش و هزینه ها کاهش پیدا کند.

- در سیستم ما ابزار اتوماسیون برای نصب و راه اندازی سیستم به عمل آورده شده که موجب شده نصب zabbix و موارد موردنیاز برروی سیستم ها راحت تر شود همچنین با به کار بردن عمل هایی از جمله raid بندی حافظه ، downtime سیستم درصورت رخ دادن مواردی همچون از دست دادن ناگهانی دیسک کاهش یابد

Customer Requirement

۱) مشتری نیازمند رابط کاربری اسان برای مشاهده وضعیت سیستم خود هست

- سیستم دارای داشبورد راحت با دسترسی مشخص و همچنین دارای مواردی همچون نشان دادن داده ها به صورت نموداری جهت درک راحت تر کاربران هست

۲) پشتیبانی از انواع مختلف کاربران تا ادمین ها و کاربران عادی و ... بتوانند از سیستم استفاده کنند

- در سیستم ما دو نقش متفاوت ادمین سیستم و کاربر زیرمجموعه ادمین با سطح دسترسی های مختلف وجود دارد تا درصورت پیچیده تر بودن شرکت بتوان دسترسی افراد متفاوت را محدود و همچنین با چنداکانت متفاوت برروی سیستم وارد شد و نظارت کرد

Architectural Requirement

۱) سیستم باید دارای معماری میکروسرویس باشد تا مقیاس پذیری و اجزا به صورت مستقل کارکنند

- در سیستم هسته اصلی برای دریافت اطلاعات سیستم از بکند و فرانت اند کاملا جدا شده اند تا درصورت نیاز برای ایجاد تغییر در نحوه دریافت اطلاعات مانیتورینگ سیستم نیازی به تغییر منطق بکند نباشد و منطق بکند بتوانند با ورژن های مختلف هسته ما کارکند

۲) سیستم باید یک مدل ارتباطی مبتنی به صف برای ارسال پیام ها برای ارسال همزمان داشته باشد

- سیستم با استفاده از موارد ارسالی همچون message broker ها این اختیار را به ما میدهد تا بتوانیم پیام هارا به صورت مدیریت شده به کاربران ارسال کنیم تا زمان اطلاع رسانی به کاربران به حداقل برسند

Structural Requirement

۱) داده های کاربران باید به صورت ایمنی ذخیره سازی شود و با توجه به نقش های هرکاربر و دسترسی هرکاربر داده ها به وی نمایش داده شود

- درسیستم ما به دلیل وجود جداول دیتابیس اطلاعات کاربر همراه با نقش و دسترسی های آن ها ذخیره شده و اطلاعات ها با توجه به دسترسی های کاربر به وی نمایش داده میشود

۲) سازماندهی داده ها به صورت سلسه مراتبی برای کاربران و کاربران زیرمجموعه فراهم شود

- درسیستم ما کاربران به صورت سلسله مراتبی ذخیره شده اند و با استفاده از رفرنس های درونی دیتابیس هرکاربر به کاربر بالادستی خود متصل میشود و توانایی تشخیص کاربران و زیرمجموعه های هرکاربر وجود دارد

Behavioral Requirement

۱) ارسال اعلان درصورت وارد شدن سیستم به حالت های اضطراری

- در سیستم به صورت اتوماتیک درصورتی که سیستم وارد شرایط بحرانی شود یا درآستانه قرارگیری در این ناحیه ها قرار بگیرد به کاربران به صورت ایمیل اطلاع رسانی میشود تا هرچه سریع تر برای رفع مشکل اقدام کنند.

۲) جلوگیری از ثبت کردن اطلاعات غلط درپروفایل کاربران

- هنگام ثبت اطلاعات مربوط به سرویس zabbix درپروفایل ، ابتدا اطلاعات بررسی میشوند که اتصال به zabbix برقرار هست یا خیر و درصورت غلط بودن اطلاعات اجازه ذخیره سازی داده نمیشود

Functional Requirement

۱) سیستم باید کشف و پیکربندی اتوماتیک برای اضافه کردن موارد فیزیکی جدید ارایه دهد

- سیستم ما به صورت اتوماتیک درصورت اضافه شدن دیسک و هارد و کارت شبکه جدید به سیستم این موارد را تشخیص و مانیتورینگ آن هارا آغاز میکند تا درصورت اضافه شدن و تغییر در اجزا فیزیکی نیازی به پیکربندی دستی نباشد

۲) سیستم باید اطلاعات را به صورت لحظه ای نمایش دهد

- سیستم ما اطلاعات را به صورت لحظه ای نمایش میدهد که این سبب میشود که اطلاعات سیستم به صورت مداوم آپدیت شوند و آخرین اتفاقات سیستم ها نمایش داده شوند تا کاربران از آخرین وضعیت سیستم خود مطلع باشند
-

Design Requirement

۱) در طراحی api موجود باید از استاندارد ها پیروی کند

- سیستم دارای api جدا برای استفاده برنامه نویس ها جهت ارتباط مستقیم با سیستم است و این api از استاندارد های موجود REST پیروی میکند

۲) سیستم باید طراحی مناسبی نسبت به سطح کاربران برخوردار باشد

- درسیستم ما با به کار گیری کنترل مبنتی بر نقش (RBAC) با تنظیمات دسترسی متفاوت این امکان را میدهد که دسترسی کاربران توسط ادمین ها قابل کنترل باشد

Derived Requirement

۱) با توجه به معماری جدا از سیستم باید شرایطی برای اتصال راحت اجزا بدهد

- اجزا سیستم به صورت جدا قرار گرفته اند و مکانیسم مناسب جهت هماهنگ سازی اجزای مختلف ایجاد شده تا اتصالات به همدیگه ایمن و پایدار باشند

۲) با توجه به ریموت بودن داشبورد باید امنیت جهت لو نرفتن داشبورد کاربران برای افراد غیرمجاز مدیریت شود

- در سیستم بااستفاده از مواردی همچون ssh و jwt token ها این مورد ارایه میشود که اطلاعات به صورت امن بین سرور و کامپیوتر جابجا شوند تا افراد غیر مجاز نتوانند به راحتی به اطلاعات کاربران دسترسی داشته باشند

Allocated Requirement

۱) سیستم احراز هویت باید توانایی کارکردن همزمان چندین کاربر را داشته باشد

- سیستم با استفاده از روش های همزمانی و بهینه سازی متفاوت این امکان را میدهد که همزمان تا ۱۰۰۰ کاربر بتوانند با حداکثر زمان 500 میلی ثانیه به سرور وارد شوند

۲) سیستم دریافت اطلاعات باید این توانایی را داشته باشد که اطلاعات موجود سیستم را به سرعت دریافت کند

- با توجه به کارکردن ریموت سیستم این تضمین را میدهد که با استفاده از روش های بهینه اطلاعات سیستم را درکمترین زمان ممکن بین سیستم درحال مانیتور شدن و سرور و سیستم ریموت مشاهده انتقال دهد

بخش دوم:

Cohesion (هم‌پیوستگی)

هم‌پیوستگی به میزان انسجام و مرتبط بودن اجزای یک ماژول یا کلاس اشاره دارد. هرچه اجزا وظایف بیشتری را با یکدیگر مرتبط و هماهنگ انجام دهند، هم‌پیوستگی بیشتر است. انواع هم‌پیوستگی به ترتیب از ضعیف‌ترین تا قوی‌ترین شامل موارد زیر است:

1. **Coincidental Cohesion (هم‌زمانی)**
اجزای ماژول کاملاً بی‌ربط هستند و به‌طور تصادفی کنار هم قرار گرفته‌اند.
   - مثال: در یک ماژول هم فایل خوانده شود، هم تاریخ سیستم چک شود، و هم پیامی در لاگ ثبت شود.
2. **Logical Cohesion (منطقی)**
اجزا از نظر منطقی به هم مرتبط هستند، اما اجرای آن‌ها مستقل از هم است.
   - مثال: ماژولی که چندین عملیات مختلف روی فایل‌ها را بسته به ورودی کاربر انجام می‌دهد.
3. **Temporal Cohesion (زمانی)**
اجزا وظایفی را انجام می‌دهند که باید در یک بازه زمانی خاص انجام شوند.
   - مثال: یک ماژول که در هنگام شروع برنامه وظایفی مثل بارگذاری تنظیمات، چک کردن نسخه و اتصال به دیتابیس را انجام می‌دهد.
4. **Procedural Cohesion (رویه‌ای)**
اجزا مراحل مختلف یک فرآیند را اجرا می‌کنند و به ترتیب به هم مرتبط‌اند.
   - مثال: ماژولی که برای پردازش یک فرم، ابتدا داده‌ها را دریافت، سپس اعتبارسنجی، و در نهایت ذخیره می‌کند.
5. **Communicational Cohesion (ارتباطی)**
اجزا بر روی داده‌های مشترک کار می‌کنند و به همین دلیل به هم مرتبط هستند.
   - مثال: ماژولی که هم داده‌های یک جدول دیتابیس را به‌روزرسانی می‌کند و هم گزارش‌هایی از آن تهیه می‌کند.
6. **Sequential Cohesion (ترتیبی)**
خروجی یک جزء، مستقیماً ورودی جزء دیگر است و این ارتباط ترتیب منطقی ایجاد می‌کند.
   - مثال: ماژولی که ابتدا داده‌ها را از فایل می‌خواند، سپس پردازش می‌کند، و در نهایت خروجی را ذخیره می‌کند.
7. **Functional Cohesion (کارکردی)**
بالاترین سطح هم‌پیوستگی است؛ ماژول فقط یک وظیفه مشخص دارد و همه اجزا برای انجام آن کار می‌کنند.
   - مثال: یک ماژول که فقط وظیفه محاسبه میانگین یک سری داده را دارد.

محاسبه Cohesion:

1. شمارش تعداد وظایف مرتبط در ماژول
2. محاسبه نسبت وظایف مرتبط به کل وظایف
3. درصدگذاری (0-100%)

Coupling (جفت‌شدگی)

جفت‌شدگی به میزان وابستگی ماژول‌ها به یکدیگر اشاره دارد. هرچه وابستگی بیشتر باشد، جفت‌شدگی بالاتر است که معمولاً مطلوب نیست. انواع جفت‌شدگی از کمترین تا بیشترین شدت وابستگی به شرح زیر است:

1. **Data Coupling (داده‌ای)**
ساده‌ترین و کمترین سطح جفت‌شدگی است. ماژول‌ها فقط از طریق تبادل پارامترهای ساده مثل اعداد یا رشته‌ها با یکدیگر ارتباط دارند.
   - مثال: تابعی که مقدار عددی را به عنوان پارامتر می‌گیرد و عملیات انجام می‌دهد.
2. **Stamp Coupling (مُهر)**
ماژول‌ها با استفاده از ساختارهای داده‌ای (مثل آبجکت‌ها یا آرایه‌ها) اطلاعات را ردوبدل می‌کنند.
   - مثال: تابعی که یک آبجکت کاربر را می‌گیرد و اطلاعات آن را تغییر می‌دهد.
3. **Control Coupling (کنترلی)**
یکی از ماژول‌ها با استفاده از پارامترهای کنترلی (مثل فلگ‌ها یا شرایط) جریان عملیات ماژول دیگر را مشخص می‌کند.
   - مثال: تابعی که یک فلگ «true/false» می‌گیرد و بر اساس آن یک عملیات خاص را اجرا می‌کند.
4. **External Coupling (خارجی)**
ماژول‌ها به پروتکل‌ها، فرمت‌های داده‌ای، یا سیستم‌های خارجی وابسته هستند.
   - مثال: یک ماژول که به یک سرویس API خارجی متصل می‌شود.
5. **Common Coupling (مشترک)**
چند ماژول به متغیرهای سراسری (global variables) دسترسی دارند. این نوع جفت‌شدگی خطرناک است، زیرا تغییر در این متغیرها بر همه ماژول‌ها اثر می‌گذارد.
   - مثال: ماژول‌هایی که همه از یک متغیر global به نام «current_user» استفاده می‌کنند.
6. **Content Coupling (محتوایی)**
شدیدترین و بدترین نوع جفت‌شدگی است. ماژول مستقیماً به محتوای داخلی ماژول دیگر دسترسی دارد.
   - مثال: یک ماژول که متدهای خصوصی یک کلاس دیگر را مستقیماً فراخوانی می‌کند.

محاسبه Coupling:

1. شمارش ارتباطات بین ماژول‌ها
2. تعیین نوع و شدت ارتباط
3. امتیازدهی (1-10)