تاکتیک های دسترس پذیری

شناسایی خطا

Ping/echo:

می توان در هنگام خرابی دستگاه های rfid reader (مثلا نرم افزار روی دستگاه هنگ کرده باشد یا در حال خواندن یک کارت گیر کند، این faultای است که امکان جلوگیری از آن وجود ندارد، پس باید اثرات failure را کمرنگ کنیم) این دستگاه ها از این روش استفاده نمود. با ارسال دستورات پینگ روی آی پی دستگاه های rifd( اگر هر دستگاه rfid reader هویتی جداگانه از طریق یک ip داشته باشد) می توان برای محاسبه ی خرابی، زمان رسیدن اطلاعات از هر دو طرف را( مثلا چه قدر طول می کشد تا پس از کشیدن کارت شماره تگ به سرور ارسال شود یا برای کانفیگ از راه دور چه قدر طول می کشد) با ارسال اطلاعات به دستگاه rfid reader دریابیم. در صورت عدم پاسخ می توان به مسئول مربوطه خبر داد تا در اسرع وقت آن دستگاه را درست کند و همچنین به افرادی که از دستگاه استفاده می کنند نیز اعلام کنیم تا بتوانند جایگزینی انتخاب کنند. Ping نمی تواند راه حل مناسبی برای این مشکل باشد چرا که ما نمی دانیم که خرابی چه زمانی رخ می دهد و اگر بخواهیم دوره ای این کار را انجام دهیم همان heartbeat می شود.

از ping می توان برای زمانی که اگر برنامه های دستگاه های موبایل ممکن است به سرور اتصال پیدا کنند اما سرور نتواند به پایگاه داده متصل شود استفاده شود تا در صورت عدم اتصال به سرور پایگاه داده خطاهای مربوطه را نشان دهیم و یا سعی کنیم از پایگاه داده دیگری بخوانیم.

Monitor:

عنصر نظارتی عنصری است که هم خطا ها را تشخیص می دهد هم مشکلات شبکه ای مانند ازدحام را نیز بررسی می کند. اما کدام قسمت از سیستم ما به این روش ممکن است نیاز داشته باشد. اگر یک جزء نظارتی برای سرور دارای نرم افزار back end داشته باشیم می توانیم حملات مانند denial of service را کنترل کنیم چنین کامپوننتی می تواند یک کامپوننت ساده مانند captcha باشد. اما اگر هدف ما کنترل تعداد کارت کشیدن ها روی دستگاه های rfid reader باشد با توجه به ماهیت سیستم که قرار است تعداد زیادی حضور اتفاق بیافتد می توان تدبیری مانند این که اگر تعداد حضور ها در روز برای یک دستگاه خاص از یک حدی بیشتر شد( تعداد افراد حاضر در دانشگاه در هر روز درون یک بازه ای است) خطایی آشکار کن. همینطور می توان تگ کارت ها را در مبدا بررسی کرد و اگر اهمیت داشت به سرور ارسال نمود. هر کارت کشیدنی(هر حضوری) برای دیگران مهم نیست.

Heartbeat:

این روش، روش خوبی است تا سالم بودن دستگاه های rifd reader را بررسی کنیم. می توان یک پیغامی به صورت دوره ای به این دستگاه ها بفرستیم (یا دستگاه rfid reader خود پیغام بفرستد) و با گرفتن پاسخ سالم بودن آن را بررسی کنیم. این پیغام می تواند بین سرور و دستگاه های rfid reader و یا برای کمتر شدن ترافیک سمت سرور از یک سیستم نظارتی داخلی استفاده شود. سپس در صورت عدم پاسخ ابتدا از دور ری استارت صورت گیرد.

Timestamp: اگر کارتی کشیده شد و قرار است افراد مربوطه از حضور آن فرد مطلع شوند، علاوه بر تاریخ حضور که ارسال می شود برای پیغام های رد و بدل شده از زمان کشیدن کارت تا فعال سازی notification حضور در نرم افزار گوشی های هوشمند، شامل گرفتن پیغام، ثبت حضور در جدول حضورها، سپس جست و جو روی گروه ها و فعال سازی notification پیغام ها داری زمان باشند تا در صورت دیر رسیدن بتوان اقدماتی صورت گیرد. مثلا اگر در زمان جست و جو در گروها ما پیغام جست و جو را دیر دریافت کنیم و این پیغام اختلاف زیادی با زمان ارسال داشته باشد به بررسی زمان حضور می پردازیم. حالت دیگری که ممکن است رخ دهد این است که پیغام حضور بعد از پیغام خروج دریافت شود. در چنین حالتی یا برای رسیدن پیغام حضور مدت زمانی صبر می کنیم یا آن را حذف می کنیم.( اگر پیغام حضور و خروج با دستگاه rfid reader را بتوان از هم تشخیص داد، اگر از gps استفاده شود این روش بهتر پاسخ می دهد).

Sanity Checking:

برای مثال پیغام حضوری به کامپوننت جست و جو روی گروه ها ارسال کنیم تا بفهمیم که آیا گروه را درست پیدا می کند یا خیر. می توان همین کار را برای کامپوننت پیشنهادات یا کامپوننت پیداکننده ی شباهت ها ارسال نمود با توجه به این که مجموعه ی حالت های ممکن را نیز می دانیم.

Condition Monitoring:

می توانیم برای ثبت حضورها حتما شرط جست و جو در گروه ها را انجام داده باشیم یعنی اگر حضور فرد مهم بود( در گروهی سر گروه بود) آن را ثبت کنیم. یا آن که در هنگام خواندن کارت به صورت محلی شماره ی سرگروه ها را که قبلا داریم چک کنیم تا از ارسال های و ثبت های بیهوده جلوگیری به عمل آوریم.

Voting: اگر دو سرور پایگاه داده داشته باشیم می بایست هر دو زمان های حضور و خروج یکسانی داشته باشند و حالت حضور( دارد یا نه) برای اعلام خبر به اعضای گروه یکسان باشد چرا که برای مثال در صورت خرابی دیگری بتواند به درستی درخواست ها را پاسخ دهد. تصور شود که یکی زمان حضور را ثبت کرده و دیگری خیر. اگر سراغ دومی برویم پس از دریافت پیغام خروج یا برای حضور صبر می کنیم یا آن را دور می ریزیم که هردو notification را غیرفعال نمی کنند. پس در چنین حالتی باید از قبل راهکاری در نظر بگیریم. می توان برای پیدا کردن شباهت ها هم نیز از دو روش مختلف استفاده کنیم تا جواب های درست تری بدست آوریم. همچنین می توان به دو روش ورودی های مختلف بدهیم(کلاس ها، مقالات، کارگاه ها، و..) و خروجی هم ممکن است متفاوت باشد. Voting به ما می گوید چه آیتم هایی شبیه تر هستند.

Exception Detection:

حالت استثنایی ما می تواند وقتی اتفاق بایفتد که پیغام خروجی(اگر پیغام خروج و ورود متمایز باشند) دریافت کردیم اما در جدول حضور های نمی توانیم حضور مربوط به آن را پیدا کنیم. در این مرحله آن را شناسایی می کنیم. می توان از time out استفاده کرد.

Self-test:

بخشی برای کامپوننت پیشنهاداتمان قرار دهیم که مجموع آیتم های مشابه ای داریم به صورت گزینشی بعضی از آن ها را انتخاب کرده به خود می دهد تا ببینیم چه پیشنهاداتی به عنوان خروجی می دهد( باقی مجموعه را باید بدهد).

Preparation & Repair

Active Redundancy: دو جدول از پایگاه داده داریم. دو دستگاه rfid reader داریم. پراسس ها برای ما اهمیتی ندارند چرا که در صورت خرابی آن ها می توان کار را از ابتدا شروع کرد و در صورتی که دوباره از اول هم شروع نکنیم مشکل فاجعه بار نخواهد بود. چون داده های ما روی هارد ذخیره می شوند. اما این روش خیلی برای سیستم ما که دسترس پذیری مهم اما بالا نیست کاربردی نمی باشد.

Passive Redundancy: می توان از این روش استفاده کرد از آن جا که داده ها حساس نیستند. در هر نیم ساعت پایگاه داده یدکی را به روز کنیم اما مشکل در زمان های حضور است که در صورت از دست رفتن باعث فعال ماندن notification می شوند.

Spare: می توان از cold نیز استفاده نمود از آن جا که دسترس پذیری بسیار بالا برای سیستم اهمیت ندارد. برای پایگاه داده مناسب نمی باشد.

Exception Handling: بعد از دریافت پیغام خروج( یک نفر کارت زده اما ما در جدول حضور ها برایش رکوردی قبلا مبنی بر حضور پیدا نمی کنیم) بدون رکورد حضور در جدول حضور، از آن جا که ماهیت پیغام خروج و حضور متفاوت نیست نمی توانیم این پیغام را شناسایی کنیم و آن را در جدول حضور به عنوان یک حضور ثبت می کنیم. برای کنترل چنین مشکلی می توان با زدن کارت یک ویژگی هم مبنی بر این که کارت زده شده برای خروج است یا خیر اضافه کنیم یا آن که در نرم افزار سمت کاربر از کاربر سوال کنیم آیا حضورش درست است یا خیر، می خواهد حضور داشته باشد یا خیر.

Rollback: roll کردن به زمانی که خروج ثبت نشده است مشکل ایجاد می کند. این روش باید به زمانی بر گردد که رکورد حضوری بدون خروج نداشته باشیم.

Software Upgrade: با اضافه کردن یک سرویس به back end باید به سرویس دهی ادامه دهیم. اگر ساختار ماژولار درستی داشته باشیم می توانیم این مشکل را برطرف کنیم.

Retry: در زمانی که از rfid reader پاسخی بر درستی خواندن دریافت نکردیم دوباره کارت را بزنیم.

Ignore Faulty Behavior:

پیغام خروجی که دریافت کردیم ( و می دانیم خروج است) اما رکورد حضور ندارد( یا حضور دیر می رسد و یا به هر دلیلی) آن را رها کنیم.

Degradation: در صورتی که نمی توانیم حضور ها را ثبت کنیم (ارتباط با پایگاه داده برقرار نمی شود و یا ...) اما می توانیم notification ها را ارسال کنیم.

Reconfiguration: اگر سرور ما نتوانست تعداد زیاد درخواست برای آیتم های مشابه را انجام دهد آن را به یک سرور دیگر ارسال کنیم ( یعنی پیدا کردن آیتم های مشابه در جای دیگر صورت گیرد از آن جا که با داده های زیادی سرو کار دارد).

Reintroduction

Shadow:

اگر بخواهیم به سیستم پیشنهادتمان ویژگی اضافه کنیم در صورت آپدیت کردن یک کپی از قبلی داشته باشیم و ورودی ها را به هر دو بدهیم سپس اگر جواب های درستی گرفتیم آن را جایگزین کنیم.

State Resynchronization:

اگر بخواهیم رکوردهای حضور را در دو سرور پایگاه داده ثبت کنیم برای هم زمانی و همگام سازی های مورد نظر روشی داشته باشیم.

Escalating Restart:

در صورت خرابی جزء ثبت حضور ها( مثلا اتصال با پایگاه داده برقرار نشود)، پیغام های حضور را در سرور ذخیره کنیم تا دوباره restart صورت گیرد . همچنین در صورت عدم اتصال rfid reader با سرور می توانیم یک سیستم محلی برای جمع آوری داده های خوانده شده از تگ ها داشته باشیم( و یا خود دستگاه خواننده هوشمند این کار را انجام دهد) و پس از برقراری اتصال این پیغام ها را دوباره ارسال کنیم. این روش کمک می کند تا کاربر احساس کند سیستم زمان های کمتری دچار خطا می شود و سیستم کامل از کار نیافتد.

Non-stop Forwarding:

در صورتی که سیستم پیشنهادات ما درست کار نمی کند، به جای آن که کل سیستم را از کار بیاندازیم یک جدول ساده از آیتم های مشابه از قبل داشته باشیم ( یک function ساده) و آن ها را به عنوان خروجی بدهیم.

Prevent Faults

Removal From Service:

در صورتی که سیستم پیشنهادات ما درست کار نمی کند آن را restart کنیم. می توان برای این کار در صورتی که مشکل از تعداد درخواست ها و کمبود فضای پردازش است آن را مجزا طراحی کنیم و امکان خاموش نمودن آن را در صورتی که از حدی فراتر رفت بدهیم و دوباره فعال سازی انجام دهیم.

Transactions:

برای آن که نوشتن های زیادی روی جدول حضور ممکن است رخ دهد پس باید کنترل آن ها و ترتیبشان را با مفاهیم و ویژگی های تراکنش کنترل کنیم.

Predictive Model:

Exception Prevention:

این که پیغام خروجی(می دانیم پیغام پیغام خروج است) ارسال شد اما رکورد حضور آن را نداریم (به دلیلی مانند قطع ارتباط با پایگاه داده) ثبت نشده است، می توانیم برای جلوگیری از این استثنا در سمت rfid reader یک رقم به رکوردهای حضور و خروج در توالی هم بزنیم (مثلا 1 نشان دهنده ی حضور و 2 نشان دهنده ی خروج )و تا زمانی آن را تا یک روز در reader نگه داریم . برای این کار می توانیم از ثبت یک بیت در تگ کارت و یا دستگاه rifd reader استفاده کنیم. همچینی می توان در سمت خواننده یا نرم افزار از کاربر بر درستی حضور و خروج سوال شود.

Increase Competence Set:

یعنی آن مجموعه ی حالت هایی که سیستم برای عمل کردن دارد. در سیستم ما اگر دستگاه rifd reader بداند اگر نمی تواند با پایگاه داده ارتباط برقرار کند یا صبر کند و یا در خود ذخیره کند یا در سرور لاگ کند تا اتصال مجدد صورت گیرد.

تاکتیک های آزمون پذیری

Control and Observe System State

Specialized Interface:

یک متد ایجاد می کنیم تا بتوانیم مقدارهای خوانده شده از دستگاه های rfid reader را گرفته و آن ها را هم لاگ کنیم و هم برای تست استفاده کنیم. یک مجموعه ی خوبی برای آزمونمان می شود. همچنین می توان یک متد reset نوشت تا تمام attributeهای کلاس پیدا کردن شباهت را به حالت پیش فرض ببریم تا بتوانیم روی آن بررسی انجام دهیم. لاگ کردن جست و جوها روی گروه ها نیز به ما کمک می کند تا بیشترین جست و جو ها و یا جست و جوهای بی نتیجه را بدست آوریم. همچنین یک متد می توان برای زمان های صرف شده از زدن کارت تا رسیدن notification به گروه ها ایجاد نمود تا بتوان محدوده ای برای آن تعریف کرد.

Record/Playback:

در هنگام ثبت ورودها و خروج ها، حالت تمام پیغام ها در گذر از اینترفیس های مختلف را نگه داریم اگر مشکلی پیش آید می توانیم در صورت بروز مشکل از این اطلاعات برای اینکه کدام قسمت ما را دچار مشکل کرد استفاده نماییم.

Localize state storage:

حالت سیستم را به صورت متمرکز با دانه بندی ریز(Localize state storage) ذخیره کنیم. می توان برای آزمون های بعدی از آن استفاده کرد. برای این کار از یک ماشین حالت استفاده می کنیم چرا که با استفاده از ماشین حالت می توانیم منبع حالات را مجزا کنیم. می توان برای حضورها از لحظه ی کارت کشیدن تا ایجاد notification حالت کل سیستم را استخراج کرده و آن را ذخیره کنیم.

Abstract data sources:

می توان از یک جدول یا فایل دارای آیتم های مشابه استفاده کرد و آن را به ماژول پیدا کننده ی شباهت داد. برای این کار باید ورودی ماژول پیدا کننده ی شباهت را بازتر انتخاب کنیم تا علاوه بر آیتم های موجود مربوط به کاربران( ذخیره شده در جداول مختلف) امکان انتخاب از یک فایل را نیز فراهم کنیم. می توان یک کلاس برای گرفتن آیتم های داده ای تعریف کرد که فانکشن های متفاوتی روی تایپ های متفاوت داشته باشد و سپس خروجی را به ماژول پیدا کننده ی شباهت بدهیم.

Sandbox:

ممکن است فردی بعد از زدن کارت به عنوان حضور فراموش کند که کارت را برای خروج بزند. بعد از ساعت بسته شدن دانشگاه به تمام این رکوردها زمان خروج می زنیم. اما برای این که این روش را تست کنیم باید تا شب صبر کنیم. پس از sandbox استفاده می کنیم که به ما اجازه می دهد هم زمان را کنترل کنیم و هم در صورت خطا سیستم را بدون هیچ دردسری را به عقب برگردانیم. یک روش آن است که که یک کلاس mock تعریف کنیم و در آن کلاک را به صورت مجازی سریع کنیم. همچنین می توانیم بعد از خواندن کارت ها، آن ها را به میل مشخصی ارسال کنیم. همچنین از آن جا که rfid reader ها در مکان دورتری از سرور قرار دارند اما درایور ها یا در سیستم محلی هستند یا می توان خواندن از آن ها را مجازی کرد و در سمت سرور ارتباطات را ایجاد کرد( اگر درایور سمت سرور باشد).

Executable assertions:

برای ورودی ماژول پیشنهاد کننده ی آیتم ها می توانیم یک assertion تعریف کنیم که ورودی های مورد نظر را چک می کند. این چک کردن هم می تواند هنگام گرفتن ورودی باشد و هم هنگام دادن پیشنهاد(خروجی ماژول)، تا نوع آیتم ها خارج از محدوده مورد نظر نباشد.

Limit Complexity

Limit structural complexity:

برای افزایش قابلیت آزمون پذیری باید پیچدیگی های ساختاری را کاهش دهیم. اگر کاربر سیستم پیشنهادی را جدای از دانشجویان و کارمندان در نظر بگیریم و ارتباط آن ها را تنها از طریق یک ماژول چک کننده ی هویت( برای مثال شماره دانشجویی و شماره شناسنامه را بخواهیم) پیچیدگی های پیاده سازی کمتر می شود و امکان گسترش نرم افزار در آینده فراهم خواهد شد. اگر از یک فریم ورک مانند mvc نیز استفاده نماییم باعث می شود که مساله ی separation of concerns نیز رعایت شود و بخش ارتباط با پایگاه داده، بخش نمایش و بخش پردازشی جدا شود تا تغییرات، اصلاحات و آزمون پذیری را نتیجه دهد. همچنین اگر چندین پایگاه داده حضور داشته باشیم، سازگاری نهایی امکان پذیر است از آن جا که داده ها حساس نیستند. اما در نهایت باید بعد از برای مثال 30 دقیقه سازگاری صورت گیرد. این در صورتی است که پایگاه داده دوم ما در اجرا دخیل نیست.

Limit nondeterminism:

از آن جا که اتفاقات سیستم پیشنهادی قابل پیش بینی هستند این تاکتیک را می توان پیاده سازی کرد. در مورد سیستم پیشنهادات نیز غیر قطعی بودن وجود ندارد و فرمول های پیشنهادی در نهایت شبیه ترین آیتم ها را انتخاب خواهند کرد.

تاکتیک های قابلیت تعامل

Locate

Discover service:

برای استفاده از سیستم دانشگاه، باید ابتدا آن را شناسایی کنیم. همان طور که گفته شده می توان اطلاعات درسی و واحد ها را از سیستم دانشگاه گرفت به جای آن که خودمان آن ها را پیاده سازی کنیم. این اطلاعات در پیدا کردن شباهت افراد کمک می کند. در اینجا می توان سرویس گرفتن این اطلاعات را با نام یا موقعیت پیدا کرد.

Manage Interfaces

Orchestrate:

باید از یک ورک فلو با کنترل پیچیدگی ها کمک بگیریم چرا که ارتباط با سیستم دانشگاه چه در تایید هویت افراد در هنگام ثبت نام و چه در گرفتن اطلاعات درسی پیغام های پیچیده ای دارد.

Tailor interface:

برای آن rfid reader هایی که ممکن است جمعیت زیادی از آن ها استفاده کنند، یک بافر برای اینترفیس مربوط به ثبت حضور قرار دهیم.

اگر به اینترفیس گرفتن واحد ها و درسهای دانشجویان در سیستم دانشگاه یک قابلیت برای تغییر نوع داده ها ایجاد کنیم تا هنگام دادن آن ها به ماژول پیدا کننده ی شباهت مشکل هرس کردن و پردازش کردن نداشته باشیم.

تاکتیک های قابلیت تغییر

Reduce the Size of a Module

Split module:

ماژول پیدا کننده ی شباهت را برای آیتم های مختلف جدا کنیم تا تغییرات در یکی ایجاد مشکل نکند.

Increase Cohesion

**I**ncrease semantic coherence:

به جای آن که از یک فانکشن شباهت معنایی در هر نوع شباهت(مانند مقالات مشابه، لینک های مشابه و....) استفاده شود بهتر است واحد شباهت معنایی را به عنوان یک ماژول جدا ایجاد کنیم و در هنگام نیاز از آن استفاده کنیم.

Reduce Coupling

Encapsulate:

برای ماژول پیدا کننده ی شباهت می توان interface ای تعریف نمود تا باقی ماژول ها در صورت نیاز از طریق این interface با آن ارتباط برقرار کنند. ارتباط مستقیم امکان پذیر نباشد.

Use an intermediary:

اگر برنامه نویسی بخواهد منبع داده ی جدیدی به سیستم اضافه کند(مثلا یک شبکه ی سنسور)، به مشکل بر می خورد. اگر واسطی تعریف کنیم که امکان گرفتن اطلاعات از منابع مختلف را فراهم کند و اجازه دهد بخش ورودی منابع از بخش تجزیه تحلیل جدا شود و بتوان منابع جدید داده را نیز اضافه کرد.

همینطور برنامه نویس بخواهد یک بخش آموزشی به سیستم اضافه کند و نیازمند سیستم ذخیره سازی مانند google drive است. باید سرویس های ذخیره سازی را برایش تعریف کنیم تا بتواند از آن ها استفاده کند. همچنین امکان اضافه کردن نیز وجود داشته باشد.

Restrict dependencies:

برای مثال برای ماژول ها سطح دسترسی تعریف کنیم تا دانشجویان و اساتید فقط ماژول های مربوط به خود را ببینند.

Refactor:

در ماژول پیدا کننده ی شباهت معنایی اگر زیرماژول هایی از آن برای منابع داده ای مختلف تعریف کنیم در صورت تغییر در کد اولیه مشکلات مربوط به تغییر کدهای مشابه از بین می رود.

Abstract common services:

همان واحد شباهت معنایی که به صورت جنرال مثلا دو کلمه را می گیرد و درصد شباهتشان را بر می گرداند. به جای آن که برای هر نوع قلم داده ای به صورت جدا تعریف شود.

Defer Binding

حالت عمومی تری به پارامترهای فانکشن ها بدهیم که بتوانند محدوده ی وسیع تری از کاربردها را پیاده سازی کنند و بتوانند در محدوده ی خارج از تعریف مقدار دهی شوند.

واحد شباهت معنایی به مقید کردن مقادیر در زمان اجرا نیاز دارد تا بتواند با ورود آیتم های جدید آن ها را نیز به عنوان پیشنهاد بدهد.

تاکتیک های کارایی

Control Resource Demand

Manage Sampling Rate:

برای دستگاه های rfid reader از Sampling Rate(تعداد کارتی که می تواند همزمان خوانده شود) در حد 3 عدد استفاده شود تا صف های طولانی ایجاد نشود.

Limit event response:

در سمت اینترفیس گرفتن داده های تگ های خوانده شده برای ثبت آن ها بهتر است از یک صف استفاده شود. با این کار تضمین می کنیم که حضورها ثبت می شوند و یا در غیر اینصورت خطای شلوغی به کاربر داده می شود.

Prioritize event:

ثبت حضورها از پیدا کردن شباهت اولویت بالاتری دارد.

Reduce overhead:

ارتباطات گرفتن با سیستم دانشگاه لازم است. این تاکتیک به کار نمی آید.

Bound execution time:

برای کارت زدن ها و ثبت آن ها در جدول حضور و پیغام دهی به کاربر در برنامه موبایل یک محدودیت اجرا و برگشت پاسخ بگذاریم.

Increase resource efficiency:

الگوریتم مربوط به پیدا کردن شباهت معنایی را بهبود دهیم و پیچیدگی را کاهش دهیم تا نیازمندی های کارایی را برآورده کند.

Manage Resources

Increase resources:

سرور قوی تری با پشتیبانی از نیازمندی های شبکه های اجتماعی(تعداد کاربران زیاد) داشته باشیم. ترافیک بالای کاربران را جوابگو باشد. در سیستم پیشنهادات نیز باید تا جای ممکن داده ها را نزدیک به هم قرار دهیم اما منابع مختلفی برای کاربران مناطق مختلف داشته باشیم تا زمان پاسخ کاهش بیابد.

Introduce concurrency:

از چند سرور برای مناطق مختلف استفاده شود یا در دانشگاه سرور محلی داشته باشیم. یا از ویژگی تِرِدها در زبان مانند جاوا برای پاسخ به درخواست ها استفاده شود.

Maintain Multiple Copies of Computations:

می توان الگوریتم پیدا کردن شباهت را بر روی دو سری داده های گرفته شده از دانشگاه (آنها که فارغ اتحصیل شده اند و آن ها که نشده اند) انجام داد. به این ترتیب بخش عظیمی از داده ها( فارغ اتحصیلان) می تواند نادیده گرفته شود و یا در صورت نیاز در سروری دیگر با قدرت بیشتر انجام شود.

Maintain Multiple Copies of Data:

می توان داده های حضور را در سروری محلی در دانشگاه نیز ذخیره کرد تا افراد حاضر در دانشگاه بتوانند با تاخیر کمتری حضورها را دریافت کنند.

Bound Queue Sizes:

صف های کارت های خوانده شده و صف های داده های گرفته شده از دستگاه های rfid reader مختلف در دانشگاه به کامپیوتر مرکزی برای ارسال به سرور باید اندازه ی مناسبی داشته باشند و این با توجه به تعداد افرادی است که ممکن است در محدودهی خاصی از زمان کارت بزنند. همچنین می توان صف های اولیت دار تعریف کرد تا بر روی rfid reader های خاصی از مکان خاصی اولویت بیشتری(مکان های شلوغ تر) بدهد.

Schedule Resources:

در هنگام لاگین کردن، به هر فرد پیشنهاداتی می شود، می توان پیشنهادات را از قبل تهیه کرد(ماتریس شباهت ها) و با ورود داده ی جدیدی (مانند یک پست جدید در قسمت سوال و جواب) باعث اجرای دوباره شود و یا برای بخش های مختلف حالت نگهداری شباهت ها و یا حالت اجرای هر باره را داشت اما اجرای هر باره نیازمند منابع زیادی است. برای مقالات با اضافه شدن شباهت مقاله ی جدید با دیگر مقالات (و همینطور ماتریس مواد داده ای دیگر) در آورده شود و در ماتریسی ذخیره شود اما اگر بخواهیم از طریق مواد داده ای خاصی مواد داده ای دیگری نیز پیشنهاد دهیم تعداد ماتریس ها زیاد می شود و پردازش آن ها زمان بر. همچنین درخواست پیشنهادات باید به سرورهای مختلف موجود نسبت داده و زمان بندی شود تا زمان صرف شده قابل تحمل باشد.

تاکتیک های قابلیت استفاده

Support user initiative

Cancel و resume و undo و aggregate:

برای آپلود مقالات و ایجاد گروه ها برای پیغام دهی در مورد حضور افراد می توان از این آپشن ها در فرم ها استفاده نمود و منابع را کنترل کرد.

عمل undo نمی تواند بر روی عمل خواندن کارت صورت بگیرد اما می توان آپشنی برای Cancel کردن حضور در نرم افزار گذاشت تا افراد بدون کارت زدن بتوانند حضور را کنسل کنند.

Pause/resume: برای دانلود مقالات و مواد درسی.

Aggregate: امکان آپلود چندین ماده ی درسی با هم فراهم باشد تا تکرار باعث خستگی کاربر نشود.

Support System Initiative

Maintain task model:

برای مثال فرمت قرار دادن لینک ها(از مواد درسی) را مشخص کنیم و در هنگام گذاشتن آن توسط کاربر به او کمک کنیم.

Maintain user model:

امکان Customization کردن گروه ها را می توان ایجاد کرد. برای مثال تعداد افراد، نحوه ی مشاهدات افراد و ...

Maintain system model:

برای مثال در کشیدن کارت در دستگاه های خواننده پیغامی مبنی بر منتظر باشید بگذاریم تا سیستم بتواند زمان ثبت حضور را کنترل کند و به کاربر اعلان کند. در آپلود مقالات و دانلود نیز از progress bar استفاده شود.

تاکتیک های امنیت

Detect Attacks

Detect intrusion:

باید از نرم افزارهای فایروال و امنیتی سمت سرور استفاده شود تا امکان جلوگیری از حملات فراهم شود.

Detect service denial:

شناسایی حملات denial-of-service که برای این کار باید گذشته ی حملات نگه داشته شود.

Verify message integrity:

در گرفتن هویت افراد در هنگام ثبت نام(باید استاد بودن و دانشجو بودن چک شود) باید از الگوریتم های رمز نگاری استفاده شود. ذخیره و مقایسه ی پسوردها نیز باید با الگوریتم های مناسب صورت گیرد. همچنین در فرم های چند مرحله ای می توان از هش کردن پیغام و ارسال به فرم بعدی استفاده کرد(مثلا فرم های بانکی، حسابداری).

Detect message delay:

برای چک کردن ارسال پیغام های دارای اطلاعات کارت های خواند شده باید زمان مناسب برآورد شود تا در صورت عملکرد نادرست بتوان دسترسی های غیرمجاز را تشخیص داد.

Resist Attacks

Identify actors:

کاربران مجاز سیستم باید برای هر بخش سیستم مشخص باشند. آن سیستم های دیگر که امکان استفاده از سیستم ما را دارند نیز باید شناسایی شوند( مثلا یک شرکت نیازمند کلان داده ی ماست). می توان لیستی ایجاد کرد و سیستم های مجاز را با کدها یا IP addresse شان در آن ذخیره نمود تا ابتدا هویت این سیستم ها چک شود.

Authenticate actors:

اگر گوشی های هوشمند اثر انگشت را پشتیبانی می کنند می توان برای بالا بردن امنیت از این روش نیز به صورت مکمل استفاده کرد. در دستگاه های rfid نیز می توان سخت افزار اثر انگشت را اضافه نمود.

Authorize actors:

برای اساتید و دانشجویان و کارمندان می توان دسترسی های مجاز تعریف کرد تا هر کدام به بخش های مجاز دسترسی داشته باشند.

Limit access:

کاربران ثبت نام نکرده می توانند بخش محدودی از سرویس ها را مشاهده کنند و فقط دانشجویان و دیگر کارمندان دانشگاه سرویس ها را دریافت کنند.

Limit exposure:

محدود کردن داده های ورودی از آن جا که می خواهیم سیستم بازی برای گسترش داشته باشیم امکان پذیر نیست. همینطور می توان در هنگام خواندن کارت های rfid به جای ارسال آنها به سرور ابتدا محدودیت چک کردن سرگروه هارا برای آنها در نظر گرفت، یعنی اگر برای کسانی حضورش مهم بود آن گاه به سرور ارسال شود.

Encrypt data:

برای شناسایی افراد از سیستم دانشگاه باید در دو سمت از کلیدی برای رمزگشایی استفاده شود تا ارتباط امن بماند و هویت افراد تضمین شود.

Separate entities:

اطاعات مربوطه به لاگین را می توان در سرور دیگری قرار داد. همچنین گرفتن اطلاعات درسی از سیستم دانشگاه می تواند از طریق سرور دیگری صورت گیرد تا دسترسی به این اطلاعات را محدود کرد.

Change default settings: در سیستم ما استفاده نمی شود.

React to Attacks

Revoke access:

در صورتی که دسترسی به اطلاعات درسی افراد گرفته شده از دانشگاه امکان پذیر شد، می توان سیستم را به طور موقت برای دانشجویان از کار انداخت.

Lock computer:

می توان از captcha در login استفاده کرد.

Inform actors:

وقتی کسی کارت می کشد و حضورش ثبت و فعال می شود به فرد میل نیز زنده شود تا از درستی حضور مطمئن شویم.

Recover from Attacks

audit trails: log گرفتن از ip کاربران و همچنین ip خواننده و نگه داشتن حالات سیستم(چه آبجکت هایی فراخوانی شده اند، به چه داده هایی دسترسی شده است و ..) می تواند به بررسی های بعدی در صورت رخداد حملات و پیدا کردن مجمرمین کمک کند.

Restore : در دسترس پذیری یکی از اهداف سرپا نگه داشتن سیستم در هنگام خرابی بود بنابرین در صورت حمله ای که سیستم به نحوی از کار می افتد باید پشتیبانی برای راه اندازی دوباره داشته باشیم می توانیم از چندین سرور استفاده کنیم.