تمرین اول درس مهندسی نرم‌افزار

گروه پنجم

اعضای گروه:

امیرحسین فراهانی 97106154
سارا آذرنوش 98170668
پیمان حاجی محمد 98170776
محمدعلی حسین نژاد 98170787
رویا قوامی 98171031

سوال اول

A:

در ادامه، با توجه به اینکه این دو اصل از لحاظ مفهومی مشابه هستند، تلاش می کنیم از پاسخ های مشابه اجتناب کنیم.

Separation concerns:

مزایا:

نگهداری آسان تر: زمانی که بخش های ایزوله شده داریم، گویا چالش ها و مشکلات جداگانه ای نیز داریم بنابراین می توان بدون اینکه هر بخش بر دیگری تاثیر بگذارد یا نگرانی ای از بابت آنها داشته باشیم، به حل هر مشکل بپردازیم.

مقیاس بندی آسانتر: تفکیک کردن، نگرانی هایی که از بابت scale کردن یک سیستم نرم افزاری وجود دارد را کاهش می دهد و حتی اگر هر یک از لایه ها، تبدیل به یک bottleneck شوند، می توان آن را به طور مستقل و بدون هیچ تاثیری بر لایه های دیگر، scale کنید.

Clean code: هنگامی که از separation concerns استفاده می کنید، در واقع بخش های کوچکتری هستند که توسعه دهنده می تواند روی تنها یک بخش مشخص تمرکز کند و بنابراین تمیزی کد بیشتر خواهد بود. به علاوه، توسعه دهندگان می توانند آن کد را با سرعت بیشتری بررسی کنند.

اجزای قابل استفاده مجدد (reusable components): این کار منجر می شود کد تبدیل به بخش های کوچکتر شده و این بخش ها در دیگر پروژه ها نیز قابل استفاده هستند.

معایب:

ادغام سخت تر: ادغام لایه های مختلف در یک سیستم می تواند چالش های زیادی به وجود آورد و هر بار باید ادغام لایه ها بررسی شوند تا مطمئن شویم که سیستم به درستی کار می کند.

توسعه گران تر: توسعه یک سیستم چند لایه منجر به ایجاد تیم های بیشتر و توسعه دهندگان بیشتر می شود، در واقع این نوع توسعه به تلاش و هزینه بیشتری نسبت به یک سیستم ساده نیاز دارد.

سیستم کندتر، عملکرد کمتر: انتقال اطلاعات بین لایه های مختلف، بیشتر از یک سیستم تک لایه طول می کشد. در یک سیستم چند لایه، ممکن است nodeهای پردازش متفاوتی داشته باشیم و داده ها نیز باید از طریق اینترنت منتقل شوند و این موضوع طبیعتا زمان بر می باشد. همچنین، نیاز داریم تا برخی فرآیند های اضافی برای انتقال داده ها بین لایه ها طراحی کنیم به طوری که این روش ها و فرآیندها قابل اتکا باشند.

Modularity:

مزایا:

تیم های ایزوله شده: توسعه دهندگان مختلف می توانند به طور مستقل روی ماژول های مختلف کار کنند، و در این روش نیازی به هماهنگی بین تیمی نداریم و می توان محصول خود را زودتر منتشر کنیم.

توسعه ویژگی‌های جدید آسان‌تر: می‌توانید ویژگی‌های جدید را بدون ایجاد اختلال در عملکرد فعلی اضافه کرد.

Encapsulation: ماژولار بودن، محیطی را فراهم می کند که می توان Encapsulation را که یکی از مهم ترین موارد در OOP است، به سادگی پیاده سازی کرد.

معایب:

ایجاد چالش در ارتباطات بین تیمی: وقتی تیم‌های زیادی با وابستگی‌های (dependency) زیاد داریم، در ارتباطات بین تیمی با چالش‌های زیادی روبرو خواهیم شد. به عنوان مثال، چالش های انتقال داده، چالش های مربوط به روابط بین فردی و غیره

طراحی معماری قابل اعتماد دشوار است: ماژولار بودن، نیازمند معماری واضح و خوب است تمام جنبه های ماژول های مختلف و وابستگی های نیز در آن در نظر گرفته شده است. این موضوع کار ساده ای نیست و به یک تیم گران پرهزینه و حرفه ای برای طراحی یک سیستم ماژولار نیاز داریم.

وابستگی های بین ماژول ها: وقتی ماژول های زیادی ساخته می شود، ممکن است وابستگی های زیادی به وجود بیاید. شرکت ها باید از این موضوع دوری کنند چراکه با این وابستگی‌ها، وقتی یک ماژول تغییر داده می شود، ممکن است روی ماژول‌های دیگر تأثیر بگذارد که منجر به مشکلات غیرمنتظره شود.

:B

سیستم های مقیاس بزرگ (Large scale systems): فرض کنید به طور مثال، یوتیب از طراحی ماژولار استفاده نکند، در این صورت، یک تیم عظیم (شاید حتی بیش از 500 نفر) مواجه هستیم و برای هر مسئله و مشکل و یا حتی هر ویژگی جدید باید تمامی این افراد با یکدیگر همکاری کنند و هماهنگی داشته باشند که چنین چیزی غیر ممکن است.

سوال دوم

رایانش ابری ارائه خدمات محاسباتی از جمله سرورها، ذخیره‌سازی، پایگاه‌های داده، شبکه، نرم‌افزار، تجزیه و تحلیل و هوشمندی از طریق اینترنت ("ابر") برای ارائه نوآوری سریع‌تر، منابع انعطاف‌پذیر و صرفه‌جویی در مقیاس است.

1) **Amazon Web Service**:

بهترین برای رایانش ابری مقیاس پذیر و انعطاف پذیر است.

امکانات:

- فرآیند ثبت نام ساده
- آسان برای استقرار
- افزودن یا حذف ظرفیت به سادگی
- دسترسی به ظرفیت نامحدود
- صورتحساب متمرکز

مزایا:

- شروع کار بسیار آسان است.
- بیش از 200 سرویس برجسته دارد.
- امکان میزبانی وب سایت های ثابت را می دهد.

- برنامه های پیچیده ای میسازد که هم مقیاس پذیر و هم انعطاف پذیر باشند.
- مزیت کلیدی AWS امنیت و قابلیت آن برای ایمن نگه داشتن اطلاعات و زیرساخت فناوری اطلاعات است. حتی در سال گذشته، حدود 90 درصد از تمام سایت‌های وردپرس هک شدند.

معایب:

- اشکالات سرویس ابری حتی امروزه در AWS بسیار رایج است.
- AWS به اتصال اینترنت شما بستگی دارد. بدون اتصال به اینترنت، شانسی ندارید.
- هنگام تنظیم ناامن سرویس ها با AWS، مهم است که این را در نظر داشته باشید.
- AWS می‌تواند داده‌های شما را استخراج کند، چیزی که ارزش نگرانی دارد.

2) Microsoft Azure:

بهترین برای طراحی و مدیریت برنامه ها از طریق یک شبکه جهانی است.

امکانات:

- طراحی، استقرار و مدیریت برنامه ها.
- از طیف گسترده ای از سیستم عامل ها، زبان های برنامه نویسی، چارچوب ها و پایگاه های داده پشتیبانی می کند.
- با ابزارهای آشنا ثبات دارد.
- کمک می کند منابع IT خود را scale کنید.

مزایا:

- مزایای اصلی این است که Azure مقیاس پذیری را ارائه می دهد. نیازی به خرید بسته های داده اضافی نیست. این به شما امکان می دهد آنچه را که نیاز دارید خریداری کنید.
- Microsoft Azure راه حل مقرون به صرفه برای بودجه IT است.
- انعطاف پذیر است، Azure به شما پیشنهاد می کند از هر چارچوب، ابزار یا زبانی استفاده کنید.
- Azure یک سیستم کنترل امنیتی به نام Detect, Assess, Diagnost, Stabilize, Close دارد که بر اساس استراتژی DADSC است. همچنین محافظت قوی در برابر از دست دادن داده ها ارائه می دهد.

معایب:

- نیاز به مدیریت و نگهداری متخصص، از جمله وصله و نظارت بر سرور دارد.
- Azure به تخصص و مهارت نیاز دارد تا اطمینان حاصل شود که تمام قطعات متحرک به درستی با هم کار می کنند.

- برای برخی از مشاغل، سرعت یک مشکل است.
- می تواند شما را مجبور کند که تمام خدماتی را که می خواهید در یک سبد قرار دهید.(It can force you to put all the services you wanted in one basket)

3) **Google Cloud Platform**:

بهترین برای ذخیره سازی و تجزیه و تحلیل داده ها است.

امکانات:

- تجسم داده ها
- مدیریت گردش کار
- صادرات/واردات داده
- گزارش جامع و آمار

مزایا:

- این یکی از بزرگترین مزایای GCP است که رقبا آن را ارائه نمی دهند. می‌توانید در طول رویدادهای تعمیر و نگهداری، انتقال مستقیم ماشین‌های مجازی را انجام دهید. عملاً هیچ زمان از کار افتادگی در خدمات یا وب سایت های شما وجود نخواهد داشت.
- بسیار مقیاس پذیر است و از مقیاس خودکار برای تنظیم خودکار مجموع نمونه های ماشین مجازی که میزبان برنامه شما هستند استفاده می کند. GCP به برنامه شما اجازه می دهد تا با مقادیر مختلف ترافیک سازگار شود.
- برنامه‌های Google Cloud می‌توانند ده‌ها هزار کاربر را به‌طور هم‌زمان بدون از دست دادن کار مدیریت کنند.
- Google Cloud به دلیل ارائه نه یک، بلکه چهار نسخه پشتیبان از داده ها مورد ستایش قرار گرفته است: ذخیره سازی خط سرد، ذخیره سازی نزدیک، منطقه ای و چند منطقه ای. هر زمان که برنامه شما یا یکی از اجزای آن با مشکل مواجه شود، پشتیبان گیری به صورت خودکار انجام می شود.

معایب:

- GCP در مقایسه با همتایان خود، مراکز داده جهانی نسبتا کمی دارد. فقط سه مرکز داده (ایالات متحده، اروپا و آسیا) دارد.
- پشتیبانی GCP زمانی که شامل رسیدگی به مسائل مشتری می شود و هزینه های پشتیبانی بسیار گران است، قوی ترین نیست.
- GCP Application Engine فقط به جاوا، پایتون، PHP و Google Go محدود می شود.

- گزینه های سفارشی سازی بسیار کمی در محصولات GCP مانند Spanner، BigQuery و Datastore موجود است. در صورت وجود هرگونه تفاوت در گردش کار با روشی که قرار است بدون هیچ وسیله ای برای بهبود عملکرد یا کشف آنچه در حال رخ دادن است استفاده کنید، مشکلات ظاهر می شوند.

4) **IBM Cloud**:

بهترین برای سیاست قیمت گذاری منعطف است.

امکانات:

- PaaS، SaaS و IaaS را ارائه می دهد.
- خدمات نسل بعدی را با استفاده از ابزارهای مختلف، مدل های داده و مدل های تحویل ایجاد میکند.
- پشتیبان گیری و بازیابی ابری
- مدیریت شبکه قوی

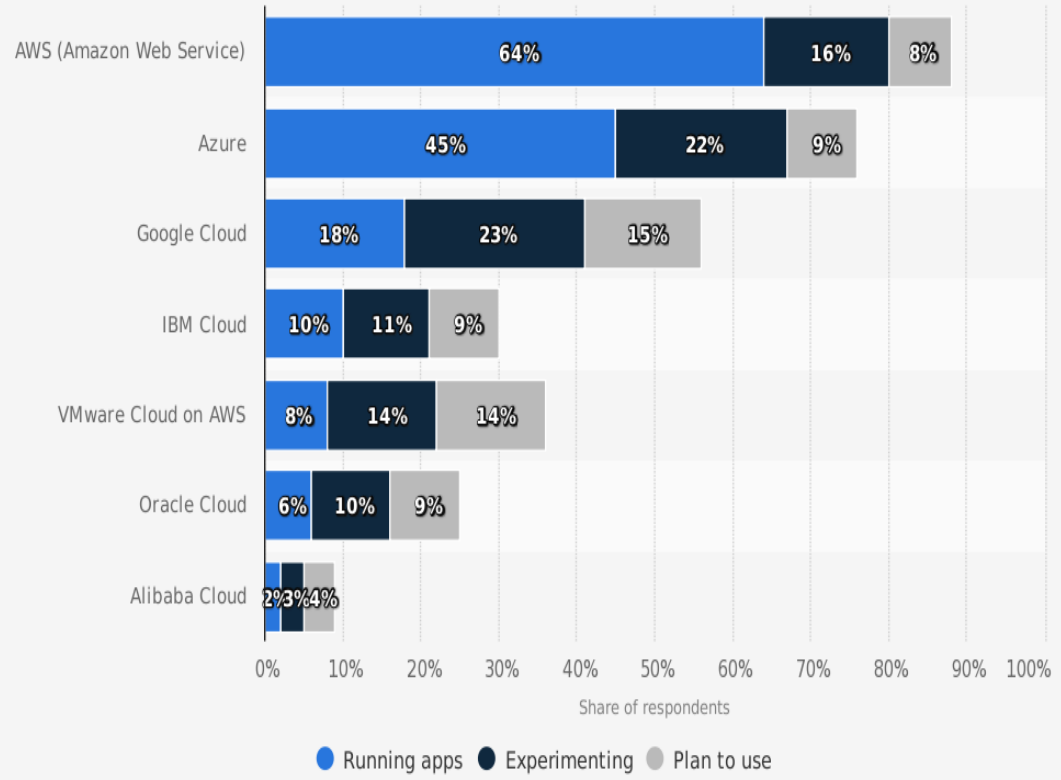
مزایا:

- به طور مداوم محصولات جدید را ارائه می دهد.
- در سیاست قیمت گذاری بسیار انعطاف پذیر است و گزینه های مختلفی برای هر بودجه ای وجود دارد.
- قدرت محاسباتی چشمگیر
- یک رویکرد داده‌ای با تقاضای بالا را اجرا میکند.

معایب:

- مراکز داده نسبتاً کمتر
- پشتیبانی مشتری چندان پاسخگو نیست.
- IBM یک شرکت بزرگ با تعداد زیادی مشتری است که ممکن است به رویکرد فردی کمتر از آنچه شما انتظار دارید منجر شود.
- اگر به دنبال یک راه حل سریع و کوتاه مدت هستید، IBM Cloud با یک سرور فلزی خالی بهترین راه حل برای شما نیست زیرا پیکربندی این سرور معمولاً فرآیندی نسبتاً وقت گیر است.

Current and planned usage of public cloud platform services running applications worldwide in 2018



Source
RightScale
© Statista 2018

Additional Information:
Worldwide; January 2018; 997 respondents; Technical executives, managers, and practitioners of cloud technologies

Figure 1. Magic Quadrant for Cloud Infrastructure and Platform Services

سوال سوم

A. رویکرد یک مفهوم کلی از سبک و ایده ای است که یک فرد برای مقابله با یک مشکل یا وضعیت خاص آن را بکار می برد. برخلاف متدولوژی، رویکرد شامل مراحل از پیش تعیین شده ای نبوده و همچنین روشی اثبات شده در طی زمان یا تجربه نمی باشد. بنابراین رویکرد هر فرد، در زمان های مختلف در مقابل یک مسئله یکسان نیز می تواند متفاوت باشد. از طرفی دیگر، متدولوژی، رویه های ساختار یافته ای هستند که بارها و بارها آزمایش شده و به اثبات رسیده اند و به عنوان ابزاری کمکی در هنگام مقابله با مسائل و مشکلات بکار می روند. همچنین یک متدولوژی مراحل گام به گام و مشخص

دارد که این ویژگی به افراد تازه کار کمک می کند تا ساده تر با مسائل خود روبه رو شوند اما در مقابل افراد با تجربه تر با رویکردها راحت تر اند زیرا مفهومی عمومی تر نسبت به متدولوژی دارند.

B. XP یکی از متدولوژی های رویکرد Agile است و همانند سایر متدولوژی های چابک، اصول اصلی رویکرد چابک همانند درگیر کردن مشتری با فرآیند تولید محصول، برقراری ارتباطات بالا درون تیمی و چرخه های تکرار شونده توسعه محصول را دارا می باشد. حال به بررسی مزایا و معایب این متدولوژی می پردازیم. چند مورد از مزایای این متدولوژی را در ادامه بررسی می کنیم:

- **صرفه جویی در هزینه و زمان:** در متدولوژی XP به دلیل آنکه تمرکز اصلی بر تحویل به موقع محصول نهایی می باشد در زمان و هزینه صرفه جویی قابل توجهی برای شرکت به ارمغان می آورد. همچنین حذف هزینه های اضافی جهت مستندسازی های بیش از اندازه و مشارکت درون تیمی بالا برای رفع مشکلات و مسائل از جمله موارد کمک کننده به این صرفه جویی ها می باشد.
- **وجود شفافیت در روند تولید:** توسعه دهندگان کارهایی که برای انجام شان متعهد شده اند در طی روند تولید پروژه قابل رهگیری و بررس است.
- **وجود همیشگی بازخورد در کار ها:** برای بهبود هر چه بهتر کار و فرآیند تولید یکی از عناصر ضروری، دادن بازخورد و انجام تغییرات لازم در مورد کارهای انجام شده و درحال انجام در جهت بهبود آن است.
- **انجام تست های مختلف همراه با فرآیند تولید محصول:** متدولوژی XP تست های معمول خود را به صورت مداوم در روند تولید محصول همواره در حال اجرا قرار می دهد تا از بروز مشکلات بزرگ در هنگام انتشار نسخه نهایی جلوگیری کرده و در زمان و پول صرفه جویی قابل توجهی صورت بپذیرد و همچنین باعث پایداری نرم افزار نیز می شود.
- **حفظ و نگهداری کارکنان با رضایت بالا:** از جمله اهداف این متدولوژی ایجاد روابط بالا و احترام بین کارکنان و شرکت می باشد و رضایت و رشد کارکنان در اولویت های اصلی این متدولوژی قرار دارد.
- **وجود ارتباط نزدیک با مشتری:** داشتن یک ارتباط نزدیک با مشتری و دخیل کردن آن در فرآیند توسعه کمک شایانی به برآورده کردن هر چه بهتر و دقیق نیاز مشتری و صرفه جویی های زمانی و مالی نیز می شود
- **انعطاف پذیری بالا نسبت به تغییر:** به دلیل ماهیت رویکرد چابک موجود در این متدولوژِی، تغییرات را به راحتی پذیرفته و در کمترین زمان آنها را اعمال می کند.

حال به چند مورد از معایب این متدولوژی می پردازیم:

- **سخت بودن ارتباطات بین تیمی در صورت دور بودن فیزیکی افراد تیم:** یکی از ویژگی های این متدولوژی تعامل بالای افراد تیم با یکدیگر است، اگر این امکان فراهم نباشد که از نظر فیزیکی تیم ها به هم نزدیک باشند این گونه تعامل به سختی انجام می پذیرد
- **تمرکز بیشتر روی برنامه نویسی تا طراحی:** عنصر مهمی که باعث فروش بهتر محصول می شود نوع طراحی آن است. اگر تمام تمرکز تیم بر روی نحوه کدنویسی نرم افزار قرار بگیرد، محصول نهایی شکل و شمایلی که مورد پسند مشتری باشد را نخواهد داشت.
- **کمبود مستند ها:** ویژگی رویکرد چابک ساده نویسی در امر مستند نویسی است. در برخی مواقع این ساده نویسی به ضرر شرکت تمام می شود. تغییرات زیاد و برخورد کردن به باگ ها در صورتی که مستند سازی نشوند با احتمال بالایی در آینده ممکن است دوباره تکرار شوند و نبود مستند نوشته شده مناسب ممکن است باعث شود دوباره برای رفع مسئله تکرار شده دنبال راه حل باشیم و این کار برای ما هزینه بیشتری بجا خواهد گذاشت.
- **بالا بودن استرس کار:** وجود ددلاین های بسیار کوتاه و فشرده باعث می شود استرس به موقع به اتمام رساندن کار بالاتر رود. برای مثال نیاز است که برنامه امروز هم کد نویسی شود و هم تست های مربوطه بر روی آن اجرا و بررسی شود.
- **نیازمند تلاش بسیار:** تطبیق زیاد و روزانه با نیاز مشتری نیازمند تلاش و زحمت روزانه زیادی می باشد.
- **نیاز به حضور مداوم مشتری در کنار کار:** معمولا مشتری ها زمان زیادی یا علاقه خاصی به ماندن در کنار کار به صورت مداوم ندارند.

متدولوژی kanban یکی از متدولوژی های رویکرد ناب (lean) می باشد. از جمله خصوصیات اصلی این متدولوژی می توان به داشتن بورد نمایش تسک ها و کارت ها اشاره کرد. بوردی که تسک های اختصاص داده شده به افراد مختلف تیم، تسک های انجام شده، در حال اجرا، و در انتظار اجرا شدن را در خود با یک نگاه به نمایش می گذارد.

حال به چند مورد از مزایای این متدولوژی می پردازیم:

- **وجود visibality در پروژه:** این متدولوژی با داشتن kanban board و kanban card ها توانسته تسک ها را به صورت واضحی برای هر کارمند تقسیم و مشخص کند. همچنین داشتن یک بورد برای بررسی وضعیت پروژه در یک نگاه کمک شایانی به مدیریت نظم و روند رشد پروژه می کند.

- **ساده بودن متدولوژی جهت استفاده:** سادگی و وجود وضوح و شفافیت بالا در فرآیند پیشرفت پروژه از جمله ویژگی های مثبت این متدولوژِی بوده که شرکت ها به سمت آن جذب می شوند
- **صرفه جویی در هزینه ها و دوری از هدر رفت منابع:** متدولوژی kanban با کنترل جریان و مدیریت خوب منابع، از هدر رفت منابع مهمی مانند زمان و پول جلوگیری می کند
- **افزایش اثربخشی و یادگیری منابع انسانی:** این متدولوژی، با توجه به چرخه ساخت و یادگیری که دارا می باشد در دراز مدت سبب رشد و پیشرفت کارکنان و افزایش قدرت تصمیم گیری آنان می گردد
- **عدم وجود بار اضافی کار بر دوش کارکنان:** به دلیل آنکه هدف این متدولوژی کار سریع در بازه زمانی کوتاه مدت بدون محدودیت حجم کاری نیست و یک محدودیت مشخصی برای حجم تسک های اختصاص داده شده وجود دارد، استرس و فشار کاری بر روی کارکنان زیاد نخواهد شد.

حال به چند مورد از معایب متدولوژی kanban می پردازیم:

- **تمرکز بر روی نتایج تا مشتری:** برخلاف رویکرد چابک که مشتری محور است، این متدولوژی بر روی فرآیند و نتیجه کار تمرکز می کند تا مشتری
- **برداشت نادرست از تسک ها:** در طی پروژه امکان دارد برداشت های نادرستی از تسک ها توسط کارکنان صورت بپذیرد، مخصوصا اگر کارمندی با روند بروز بروز پروژه پیش نرفته باشد.
- **محدودیت در تعداد تسک های ناکامل:** گاهی اوقات این نوع محدودیت در تعداد تسک های انجام نشده باعث مشکل در فرآیند تیم می شود، از جمله این مشکلات ورود تسک های جدید با اولویت بسیار بالا بعد از تعداد زیادی تسک انجام نشده با اولویت کمتر می باشد.
- **نبود دوره زمانی مشخص و بروز تاخیر ها:** عدم وجود دوره زمانی مشخص برای به اتمام رساندن تسک ها در این متدولوژی باعث به وجود آمدن تاخیر در فرآیند توسعه نرم افزار می شود.
- **نامناسب در محیط و سیستم های با پویایی بالا:** فرضی که در مورد کار ها و تسک ها در این متدولوژی شده است ایستا بودن سیستم و تسک ها می باشد و این متدولوژی برای محیط های پویا مناسب نمی باشد.

C. همانطور که در بخش قبل به فواید و مضرات هر دو متدولوژی اشاره شد، متدولوژی XP در پروژه هایی که نیازمند انعطاف پذیری و سازگاری بالا هستند و همچنین همواره در تعامل نزدیک با مشتری

برای اعمال تغییرات لازم به صورت مداوم هستند مناسب تر است. از طرف دیگر، متدولوژی Kanban در پروژه های با جریان کاری ثابت بهتر عمل می کنند و همچنین در این پروژه ها دید یا به اصطلاح visibality بالایی را بر روی روند پیشرفت پروژه مهیا می کند. علاوه بر این ها، این متدولوژی مناسب پروژه های با نیازمندی بهبود به صورت مستمر می باشد.

سوال چهارم

A. مفهوم فرد، به اشخاص مختلف که عضو تیم هستند اشاره میکنند. که این افراد مسئولیت‌ها و وظایف مختلفی به عهده دارند. در مقابل، در هر تیم نقش هایی مانند مهندس نرم‌افزار، توسعه دهنده فرانت‌اند، QA و ... تعریف میشود. هر نقشی، مسئولیت‌ها و وظایف مشخص خود را دارد. و هر یک از افراد تیم میتواند به صورت همزمان نقش های مختلفی در یک تیم را ایفا بکند. برای مثال ممکن هست که QA و Front-End Developer که دو نقش مجزا هستند، به یکی از اعضا منتسب شوند.

B. **مالک محصول:** با توجه به اینکه در اسکرام یکی از اهداف تغییر سریع میباشد،باید نیازهای مشتری نیز در اولویت قرار گیرند. این کار وظیفه مالک محصول میباشد. مالک محصول مسئول به حداکثر رساندن ارزش محصول توسعه یافته توسط تیم اسکرام است و نیازهای مشتری یا کاربر نهایی را نشان می دهند. همینطور مالک محصول مسئول ایجاد و نگهداری بک لاگ محصول هست، که شامل لیست اولویت بندی شده ای از ویژگی ها و تسک ها میباشد.

اهمیت: مالک محصول از نزدیک با تیم توسعه همکاری می کند تا اطمینان حاصل کند که محصول به گونه ای توسعه می یابد که نیازهای مشتری را برآورده کند، و آنها اطمینان می دهند که تیم روی مهم ترین موارد موجود در بک الگ محصول کار می کند.

اسکرام مستر: وظیفه وی، اطمینان از اجرای درست و دقیق فرایند اسکرام، و اطمینان از اینکه تیم به درستی از اصول اسکرام پیروی میکند میباشد.همینطور از اجرای مناسب جلسات مختلف اسکرام مانند دیلی ،پلنینگ، ریویو و رترو اطمینان حاصل میکند. نقش دیگر اسکرام مستر، بهبود عملکرد و تاثیر همکاری صاحب محصول و تیم توسعه میباشد.

اهمیت: اسکرام مستر، سعی میکند موانع سر راه تیم توسعه را برطرف کند تا توسعه محصول با بیشترین بازده اتفاق بیافتد. همینطور اصلی ترین نقش وی، نظارت به اجرای دقیق متدها و قوانین اسکرام میباشد.

تیم توسعه: این تیم که یک تیم cross-functional و self-organizing هست، وظیفه دارد یک increament قابل عرضه را تا انتهای اسپرینت برساند. تیم توسعه با مالک محصول همکاری می

کند تا نیازهای محصول را درک کند و آنها با هم کار می کنند تا وظایف مورد نیاز برای تکمیل هر مورد در بگ‌لاگ محصول را شناسایی کنند.

اهمیت: این تیم با همکاری که با مالک محصول دارد، featureهایی را طبق نیازهای مشتری توسعه میدهد و خب انجام کارهای توسعه بر عهده این تیم میباشد.

سوال پنجم

A.

توسعه تست محور (Test Driven Development یا TDD):

TDD یک روش توسعه نرم افزار است که در آن تست ها قبل از توسعه کد نوشته می شوند. در این روش به صورت افزایشی به توسعه نرم افزار می پردازیم به طوری که در TDD، ابتدا تست‌های کوچکی نوشته می‌شوند که کوچک‌ترین افزایش ممکن در عملکرد را آزمایش می‌کنند. سپس، کد نوشته می شود تا آن تست نوشته شده را پاس کند و به دنبال آن، کد مجدداً refactor می‌شود تا کارآمد و قابل نگهداری شود. این چرخه آزمایش، کد نوشتن و refactor کردن، به طور مداوم تکرار می شود. در واقع TDD تضمین می کند که کد به طور کامل آزمایش شده است و تغییرات جدید عملکرد موجود را خراب نمی کند.

در واقع در این روش هدف این است که اطمینان حاصل کنیم که کد توسط تست ها پوشش داده شده است و تست ها نیز به خوبی تعریف شده اند و همچنین اطمینان حاصل می کنند که کد همانطور که انتظار می رود کار می کند.

مزایا:

- منجر به تولید کدهای ماژولارتر و قابل نگهداری تر می شود.

- به شناسایی خطاها در اوایل چرخه توسعه کمک می کند، که هزینه رفع خطاها را در آینده کاهش می دهد.

- به توسعه دهندگان کمک می کند تا روی نوشتن کدی که requirementها را برآورده می کند تمرکز کنند.

بازسازی مجدد (Refactoring)

Refactoring، فرآیند بهبود ساختار و طراحی کد موجود، آن هم بدون ایجاد تغییر رفتار در آن است. این فرآیند شامل ایجاد تغییرات کوچک و تدریجی در کد برای بهبود خوانایی، قابلیت نگهداری

(maintainability) و عملکرد (performance) است. این فرآیند بخشی ضروری از روش XP است، زیرا به کاهش technical depth و افزایش کیفیت کد کمک می کند.

مزایا:

- کیفیت کد را با افزایش خوانایی و maintainability بهبود می بخشد.

- به شناسایی و از بین بردن code smells کمک می کند.

- بدهی فنی (technical depth) را با پاکسازی کد موجود کاهش می دهد.

توضیح اصطلاحات استفاده شده در بالا:

Technical depth: بدهی فنی (همچنین به عنوان بدهی فناوری یا بدهی کد شناخته می‌شود) زمانی رخ می دهد که تیم های توسعه اقداماتی را برای تسریع در تحویل یک بخش از عملکرد یا پروژه را انجام می دهند که بعدا نیازمند refactor کردن خواهد بود. به عبارت دیگر، نتیجه اولویت دادن تحویل سریع به کد با کیفیت بالا است.

Code smells: به زبان ساده، بوی کد نتیجه برنامه نویسی ضعیف یا نادرست است که می معمولا مستقیما مربوط به خطایی است که توسط برنامه نویس در طول فرآیند کدنویسی ردیابی می شود و معمولا ناشی از عدم نوشتن کد مطابق با استانداردهای لازم است.

یکپارچه سازی پیوسته (CI یا Continuous Integration):

CI یک روش توسعه نرم افزار است که در آن توسعه دهندگان کد خود را به طور مکرر و معمولاً چندین بار در روز در یک repository مشترک ادغام می کنند. هر یکپارچه سازی (integration)، قبل از اینکه داخل ریپازیتوری مرج شود، توسط یک فرآیند خودکار build کردن و test کردن، تأیید می شود تا مشکلات زود تشخیص داده شوند و بتوان آنها را برطرف کرد. CI تضمین می کند که کد همیشه قابلیت release کردن دارد و به کاهش مشکلات یکپارچه سازی نیز کمک می کند. هدف CI شناسایی خطاها در اوایل چرخه توسعه است، آن هم قبل از اینکه برطرف کردن آنها هزینه بیشتری در پی داشته باشد.

مزایا:

- به شناسایی خطاها در اوایل چرخه توسعه کمک می کند.

- همکاری بین اعضای تیم را بهبود می بخشد.

- با اطمینان از اینکه همه تغییرات در یک repository ادغام می شوند، خطر مشکلات یکپارچه سازی را کاهش می دهد.

Pair Programming:

Pair Programming یک تکنیک توسعه نرم افزار است که در آن دو توسعه دهنده با هم روی یک codebase کار میکنند. یک توسعه دهنده کد را می نویسد و توسعه دهنده دیگر آن را بررسی میکند و بازخورد ارائه می دهد. این دو توسعه دهنده اغلب نقش ها را تغییر می دهند و این سبک برنامه نویسی به بهبود کیفیت کد، کاهش عیوب و انتقال دانش بین اعضای تیم کمک می کند

مزایا:

- به شناسایی خطاها در اوایل چرخه توسعه کمک می کند.

- به اشتراک گذاری دانش بین اعضای تیم را تسهیل می کند.

- کیفیت کد را با ارائه بازخورد فوری بهبود می بخشد.

بازی برنامه ریزی (Planning Game):

بازی برنامه ریزی تکنیکی است که در روش XP برای برنامه ریزی و پیش بینی پروژه استفاده می شود. در این روش مشتری و تیم توسعه با هم کار می کنند تا ویژگی های مورد نیاز پروژه را شناسایی و اولویت بندی کنند. سپس تیم تلاش لازم برای پیاده سازی هر ویژگی را تخمین می زند و مشتری اولویت خود را تعیین می کند. این رویکرد مشارکتی تضمین می کند که اهداف و الزامات پروژه برای همه افراد درگیر روشن است و تیم ابتدا روی ارزشمندترین ویژگی ها کار می کند.

مزایا:

- کمک می کند تا از اینکه تیم توسعه ابتدا روی مهمترین ویژگی ها کار کند اطمینان حاصل شود.

- با تعیین اولویت های واضح به مدیریت scope creep کمک می کند.

- ارتباط بین تیم توسعه و مشتری را بهبود می بخشد.

B.

توسعه تست محور (Test Driven Development یا TDD):

سناریو : فرض کنید برنامه نویسی هستیم که وظیفه اضافه کردن قابلیتی جدید را به کد پروژه داریم (مثلا این قابلیت که هنگام ورود کاربر، اطلاعات صحیح نشان داده می شود یا خیر). برای اینکه توسعه به صورت TDD باشد، ابتدا باید یک تستی که در صورت نبودن آن قابلیت شکست می خورد می نویسیم (failing test) و سپس حداقل مقدار کد لازم برای قبولی این تست را می نویسیم. این فرایند نوشتن تست و کدنویسی را مکررا ادامه می دهیم تا زمانی که تمام این قابلیت و ویژگی هایش پیاده سازی شده باشد و تمامی تست ها پاس شوند. بعد از این کار می توان کد را تغییر دهیم تا خوانایی و نگهداری آن بهبود یابد و دوباره برای هر بخش از عملکرد این قابلیت، تست ها را تکرار می کنیم.

بازسازی مجدد (Refactoring)

سناریو: فرض کنید تیمی هستیم که چند ماه است که روی پروژه‌ای کار کنیم و متوجه شدیم که کد پروژه به طور فزاینده‌ای پیچیده شده نگهداری آن نیز دشوار است. به علاوه، بخش‌های زیادی از کد وجود دارد که اضافی یا تکراری شده‌اند، و همچنین بخش‌های مختلفی از سیستم وجود دارد که آن‌طور که باید کار نمی‌کنند. در این سناریو می توان با استفاده از تکنیک های refactoring برای بهبود این موارد استفاده کرد بدون اینکه رفتار کد تغییر کند. به عنوان مثال، تیم می تواند مناطقی از کد را که اضافی هستند شناسایی کرده و آنها را در یک تابع واحد ادغام کند. همچنین می تواند قسمت هایی از کد که عملکرد خوبی ندارد را شناسایی کرده و آن را برای بهبود عملکرد بهینه کند. از طریق این فرآیند، تیم می‌تواند به تدریج کیفیت پایگاه کد را بهبود بخشد و نگهداری آن را در طول زمان آسان‌تر کند.

بعد از استفاده از این تکنیک ها، می توان آن را تست کرد که از کارکرد درست کد مطمئن شد.

یکپارچه سازی پیوسته (CI یا Continuous Integration):

سناریو: فرض کنید در شرکتی در حال کار بر روی یک محصول نرم افزاری جدید هستیم و با اعضای تیم تصمیم گرفته ایم که از CI برای کشف خطاها در اوایل چرخه توسعه استفاده کنیم. بنابراین هر یک از ما، به طور منظم کد خود را در یک ریپازیتوری مشترک ادغام می کنیم و تست های خودکار نیز روی هر ادغام انجام می شوند تا مطمئن باشیم کد به درستی کار می کند. باید این فرآیند را در طول چرخه توسعه ادامه دهیم تا مطمئن شویم محصول نهایی، عاری از هرگونه خطایی است.

Pair Programming:

سناریو: فرض کنید شما و همکارتان دو برنامه نویس هستید که روی ویژگی جدیدی برای وبسایت شرکت کار میکنید و تصمیم می گیرید از فرآیند pair programming برای بهبود کیفیت کدی که میزنید استفاده کنید. بنابراین به طور نوبتی یکی از شما باید مسئول زدن کد شود و دیگری برای آن بازخورد ارائه کند و سپس ترتیب عوض شود. در مورد مسائل و مشکلاتی که با آن برخورد می کنید نیز، باید با یکدیگر صحبت کرده و برای یافتن راه حل با یکدیگر همکاری کنید. در نهایت پس از تکمیل این ویژگی جدید باید هر دو کد را بررسی و تست کنید تا مطمئن شوید requirementها رعایت شده است و خطایی در کد پیاده سازی شده وجود ندارد.

بازی برنامه ریزی (Planning Game):

سناریو: فرض کنید تیمی هستید که در حال کار بر روی یک پروژه نرم افزاری جدید برای یک مشتری می باشید و تصمیم دارید از روش planning game استفاده کنید تا اطمینان حاصل کنید که مهمترین ویژگی ها در ابتدای کار توسعه پیدا می کنند. بنابراین تیم شما باید با مشتری همکاری کند تا ویژگی هایی که باید توسعه داده شوند اولویت بندی شود و سپس تیم شما باید به طور مرتب طرح را با مشتری بررسی کرده تا از همسو بودن نیازهای او و آنچه در حال پیاده سازی است اطمینان حاصل کنید. این فرآیند را در طول چرخه توسعه باید آنقدر ادامه دهید تا مطمئن باشید محصولی که به مشتری ارائه می دهید نیازهای او را برآورده می کند.

سوال ششم

مدل توسعه نرم افزار (software development model):

مدل توسعه نرم افزار چارچوبی است که مراحل توسعه نرم افزار را مشخص می کند که یک رویکرد ساختاریافته برای فرآیند توسعه نرم افزار، از مرحله جمع آوری نیازهای اولیه تا استقرار (deployment) و نگهداری (maintenance) ارائه می دهد. نمونه هایی از مدل های توسعه نرم افزار شامل مدل آبشار، مدل چابک، مدل اسپیرال و مدل V است.

متدولوژی توسعه نرم افزار (software development methodology):

متدولوژی توسعه نرم افزار مفهوم گسترده‌تری است که رویکرد، اصول و شیوه‌های مورد استفاده در توسعه نرم‌افزار را در بر می‌گیرد. که شامل ابزارها، تکنیک ها و فرآیندهای مورد استفاده برای مدیریت چرخه عمر توسعه نرم افزار است. نمونه هایی از متدولوژی های توسعه نرم افزار عبارتند از Scrum، Kanban، Lean و XP (Extreme Programming).

تفاوت این دو:

تفاوت اصلی بین مدل‌های توسعه نرم‌افزار و متدولوژی‌های توسعه نرم افزار در این است که مدل‌ها، رویکرد گام به گام‌تری برای توسعه نرم‌افزار ارائه می‌دهند، در حالی که متدولوژی‌ها انعطاف‌پذیرتر و سازگارتر هستند. مدل‌ها تغییر پذیری کمی دارند و ممکن است برای همه انواع پروژه‌ها یا سازمان‌ها مناسب نباشند. از سوی دیگر، متدولوژی ها به گونه ای طراحی شده اند که انعطاف پذیرتر باشند و می توانند متناسب با نیازهای پروژه ها و سازمان های مختلف، customize شوند.

سوال هفتم

A. همانطور که می دانید، مدل حلزونی در پروژه های با ریسک بالا و نیازمندی های زیاد و پیچیده مورد استفاده قرار می گیرد. سناریو ذکر شده نیز به دلیل حساسیت بالا و پیچیدگی هایی که دارد، مدل حلزونی مناسب ترین مدل ایجاد نرم افزار برای آن خواهد بود. علاوه بر آن در سناریو ذکر شده که پژوهشگران ایده های مبهمی برای پیشبرد پروژه دارند و برنامه ریزی درستی برای آن وجود ندارد، در این گونه سناریو ها نیز مدل حلزونی بخاطر ساختار iterative و تحلیل و بررسی ریسک و نیازمندی ها و بهبود کیفیت به صورت مداوم، مناسب می باشد.
B. این سناریو دو ویژگی مهم دارد.
   a. اول اینکه شرکتی که قرار هست توسعه را انجام دهد، تجربه کافی ندارد و نیازهای کامل سیستم و جزئیات پیاده سازی در اول کار مشخص نیستند. در نتیجه با توسعه به صورت iterative میتوان عدم قطعیت طراحی، و کمبود اطلاعات را جبران کرد. علاوه بر این، بایستی در هر فاز از توسعه این محصول، ریسک‌هایی که وجود دارد بررسی شود تا عدم قطعیت‌های اولیه، با ضرر حداقلی کنترل شوند.

b. در ابتدای کار صرفا نیاز به پیاده سازی سیستم برای برخی از ساختمان‌ها وجود دارد. درواقع با توسعه به روش Incremental میتوان با توسعه محصول کوچک،و افزدون ویژگی‌های مختلف به محصول نهایی رسید.

با توجه به سه ویژگی عدم‌قطعیت و نیاز به تحلیل ریسک، مواجه شدن با نیازمندی‌های جدید در توسعه محصول، نیاز به توسعه محصول ابتدایی و توسعه بیشتر در ادامه میتوان گفت مدل حلزونی میتواند مدل مناسبی برای این سناریو میباشد چرا که هم ویژگی incremental و iterative بودن را دارد و همینطور فازی برای تحلیل ریسک دارد.

C. با توجه به سناریو، شرکت نرم‌افزاری احتمالا یک خط تولید نرم‌افزار دارد که برای شرکت‌های مختلف، نرم افزاری که قبلا توسعه داده را شخصی سازی میکند. در واقع این شرکت نیازها، و ساختار نرم افزار را درک کرده و برای شخصی سازی، تعدادی ویژگی خاص مورد نیاز آن شرکت را باید پیاده سازی نماید. درواقع بخش اعظمی از طراحی و پیاده‌سازی محصول انجام شده. حال با توجه به این موارد متدولوژی مورد نظر میتواند یکی از متدلوژی های Incremental از جمله Agile باشد. چرا که با توجه به شرایط نیازهای جدید مشتری مطرح میباشد و نیازی به طراحی دوباره محصول نمیباشد.