به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

درس مهندسی نرم افزار ۲، نیمسال دوم سال تحصیلی ۱۴۰۲-۱۴۰۱

تمرین یک

توضیحات تکمیلی:

- مباحث این تمرین از چهار فصل اول کتاب پرسمان و اسلایدهای ماهیت نرم افزار، فرایند توسعه و فرایند یکپارچه می‌باشد.
- مهلت تحویل تمرین **۱۹ اسفند** در نظر گرفته شده است.
- پاسخ به تمرین‌ها به صورت **گروهی** می‌باشد و همه اعضای گروه موظفند در آن مشارکت داشته باشند.
- در صورت برخورد با پاسخ های مشابه بیش از حد بین گروه های مختلف، نمره کسر خواهد شد.
- ارسال فایل مربوطه در سامانه کورسز توسط سر گروه(نام ذکر شده در گوگل شیت گروه بندی های درس) کافی می‌باشد و نیازی به بار گذاری توسط تمام اعضای گروه نیست.
- نظم و خوانایی تمرین از اهمیت بالایی برخوردار می‌باشد.
- خواهش می‌شود تمرین خود را در قالب یک فایل PDF با نام"**SE2_HW[number]_[Group Number]**" مانند: "SE2_HW1_6.pdf" در مهلت یاد شده در سایت درس بار گذاری فرمایید.
- پرسش‌های خود درباره این تمرین را می‌توانید از راه ایمیل se2springta@gmail.com بیان کنید.

1 .اصول کلی هوکر[1] را نوشته و هریک را به اختصار توضیح دهید.

:The Reason It All Exist

هدف اصلی وجود یک سیستم، ارائه یک سرویس به کاربر است. همه تصمیمات برای توسعه محصول، باید با این پرسش سنجیده شوند که آیا این ویژگی/نیازمندی /روش توسعه/پلتفرم مدنظر، سیستم ما را برای کاربر ارزشمندتر میکند. با قی اصول بر مبنا این اصل هستند و تنه ا در صورتی که پاسخ سوال باال "بله" باشد، باید آن را محور ادامه توسعه قرار داد.

:Kiss

طراحی نرم افزار یک فرایند تصادفی نیست. در هر طراحی باید به موارد زیادی توجه شود. همه طراحی ها باید تا حد امکان ساده باشند، اما نه بیشتر از آن. این باعث میشود سیستمی با درک ساده تر و نگهداری آسانتر ایجاد شود .این به این معنی نیست که ویژگی ها، حتی ویژگی های داخلی، به خاطر سادگی حذف شوند. در واقع، هرچه طراحی ساده تر، بهتر و زیباتر است. ساده همچنین به معنای "سریع و کثیف" نیست. بلکه در چندین iteration زمان و کار زیادی می برد تا سیستم ساده باشد که حاصل این تلاش، نرمافزاری است که نگهداری آن آسانتر و دارای خطای کمتر است.

:maintain the vision

داشتن دیدی واضح، برای موفقیت یک پروژه نرمافزاری ضروری است. بدون داشتن آن، پروژه تقریباً به طور معمول به "دو یا چند ذهنیت" درباره خودش ختم می شود. بدون یکپارچگی مفهومی، یک سیستم تهدید می کند که به یک شبکه پارچه ای از طرحهای ناسازگار تبدیل شود که با پیچ های اشتباه به هم چسبیده شده اند. فرایند مذاکره بر سر دید معماری یک سیستم نرم افزاری، حتی سیستم های طراحی شده ی خوب را نیز ضعیف و در نهایت شکست می دهد. داشتن یک معمار قدرتمند که می تواند چشم اندازی را حفظ کند و انطباق را به اجرا بگذارد، میتواند موفقیت یک پروژه نرم افزاری را تضمین کند.

:What you produce, others will consume

نرم افزارهای صنعتی قدرتمند به ندرت در یک محیط خلوت ساخته و استفاده می شوند. به بیانی دیگر، کسی دیگر از سیستم شما استفاده، نگهداری و مستندسازی می کند یا به طور کلی وابسته به فهم سیستم شما است. بنابراین، همیشه در هنگام طراحی و پیاده سازی ، باید بدانید که دیگران نیز باید بتوانند آنچه انجام می دهید را فهمیده و از آن استفاده کنند. مخاطبان هر محصول توسعه نرم افزار، بسیار زیادند. بنابراین در مشخص کردن نیازمندی های محصول، به نظر کاربران باید توجه شود. در طراحی، باید به فکر اجراکنندگان باشید. در کدنویسی، باید به کسانی که باید سیستم را نگهداری و گسترش دهند، توجه کنید. کسی ممکن است بخواهد کدی که شما نوشته اید را دیباگ کند و این باعث می شود که او نیز یک کاربر از کد شما باشد. راحت کردن کار کاربران، ارزشی زیادی به سیستم شما اضافه می کند.

:Be open to future

[1] Hooker’s General Principles

یک سیستم با عمر طولانی، ارزش بیشتری دارد. در محیط های محاسباتی امروزی که نیازمندی ها به سرعت تغییر می کنند و پلتفرم های سخت افزاری که به تازگی تولید شده در عرض چند ماه منسوخ می شوند، طول عمر نرما فزارها به طور معمول در ماه ها به جای سال ها اندازه گیری می شود. با این حال، سیستم های نرم افزاری "صنعتی قدرتمند "باید برای مدت زمان طولانی تری باقی بمانند. برای داشتن موفقیت در این زمینه، این سیستم ها باید آمادگی برای سازگاری با تغییرات و دیگر تغییراتی که ممکن است در آینده رخ دهند، داشته باشند. سیستم هایی که با موفقیت این کار را انجام می دهند، آن هایی هستند که از ابتدا به این شکل طراحی شده اند. هرگز خودتان را در گوشه ای از طراحی گرفتار نکنید. همیشه بپرسید "چه میشود اگر فلان اتفاق بیفتد"، و با ایجاد سیستم هایی که مسئله کلی را حل می کنند، نه فقط مسئله خاص را، برای تمام پاسخ های ممکن آماده باشید. این می تواند به استفاده ی مجدد از کل سیستم نیز منجر شود.

Plan ahead for reuse:

استفاده مجدد، در مصرف زمان و انرژی صرفه جویی می کند. رسیدن به سطح بالایی از استفاده مجدد بدون شک دشوارترین هدف در توسعه یک سیستم نرم افزاری است. استفاده مجدد کد ها و طرح ها به عنوان یکی از مزایای استفاده از فناوري هاي شیءگرا اعلام شده است. با این حال، بازگشت سرمایه در این زمینه به طور خودکار اتفاق نمی افتد. براي بهره برداری از امکانات استفاده مجدد که برنامه نویسی شیءگرا (یا سنتی) فراهم می کند، نیاز به پیش بینی و برنامه ریزي داریم. در هر سطح از فرایند توسعه سیستم، روش هاي زیادی براي استفاده مجدد وجود دارد. برنامه ریزي براي استفاده مجدد، هزینه را کاهش داده و ارزش اجزاي قابل استفاده مجدد وسیستم هایی که آنها در آن استفاده می شوند را افزایش می دهد.

Thinking:

این آخرین اصل، احتمالاً بیشتر نادیده گرفته شده است. قرار دادن تفکری روشن و کامل پیش از اقدام تقریباً همیشه نتایج بهتری را به دنبال دارد. وقتی شما درباره چیزی فکر می کنید، احتمالاً به درستی انجامش می دهید. همچنین دانشی در مورد روش صحیح انجام کار نیز به دست می آورید. اگر شما درباره چیزی فکر کنید و باز هم اشتباه کنید، آن اشتباه به عنوان یک تجربه ارزشمند محسوب می شود. وقتی تفکری روشن و کامل به یک سیستم اختصاص داده شده باشد، ارزش آن افزایش می یابد. اعمال شدن اصول اول تا ششم، به فکر کردن شدید نیاز دارد و به کار گرفتن این اصول پاداش های بی شماری به دنبال دارد. اگر تمامی مهندسان نرم افزار و تیم های نرم افزاری این هفت اصل هوکر را دنبال کنند، بسیاری از مشکلاتی که در ساخت سیستم های پیچیده تجربه می کنیم، حذف خواهند شد.

2. الف) با توجه به اینکه امروزه نرم افزار ها در حال رشد هستند، آیا مدل های فرایند سنتی نظیر آبشاری[2] را مناسب برا تولید و توسعه نرم افزار میدانید؟ چرا؟

خیر،

امروزه نرم افزارها برای بقای خود و ارائه سرویس بهتر دائما با نیازمندی های متفاوت و جدیدی روبرو میشوند. بنابراین برای توسعه محصولات نرم افزاری باید از روش هایی استفاده کنیم که ایجاد تغییرات در آنها راحت تر باشد. شیوه های سنتی و آبشاری برپایه یک نیازمندی و فرضیه ثابت پایه ریزی شده اند و ایجاد تغییرات در اینگونه ساختارها بسیار هزینه بر و مشکل می باشد. از این رو برای توسعه نرم افزاری

[2] Waterfall

کارآمد باید از روشی استفاده کنیم که ایجاد تغییرات در آن کم هزینه تر باشد. همچنین در فرایندهایی مانند فرایند آبشاری محصول دیر آماده تحویل میشود و در بازار پرتحرک امروزی این امر باعث از دست دادن مشتری و بازار فروش می شود.

ب) بنظر شما در چه سیستم هایی مدل آبشاری جوابگو است؟ آنرا نام ببرید و دلیل خود را ذکر کنید؟

فرایند آبشاری برای سیستم هایی مناسب است که نیازمندی های آنها به طور کامل در ابتدا مشخص باشد و تغییرات احتمالی در طول فرایند توسعه بسیار کم باشد. همچنین این سیستم ها نیازی به ارائه سرویس به صورتی افزایشی نداشته باشند. در چنین شرایطی می توان از فرایند آبشاری استفاده کرد.

-سیستم هایی که در آنها تست و کنترل کیفیت بسیار اهمیت دارد زیرا در فرایند آبشاری می توان به صورت ساختارمند فرایند تست تعریف کرد و تا پاس نشدن تست ها به مراحل بعدی نرفت.

-سیستم هایی که نیاز به طراحی و دقیق و بدون نقص جهت پاسخگویی به تمام نیازمندی ها دارند.

-سیستم هایی که نیازمندی های آنها در ابتدا به صورت شفاف بیان شده است و تغییرات ناچیزی دارند مانند پرداخت کارت به کارت که از استاندارد ISO ۸۵۹۰-استفاده می کند و سالیان سال است ثابت است.

۳. الف) مدل spiral و مدل prototype را با یکدیگر مقایسه کنید و مزایا و معایب هریک را ذکر کنید.

ب) فرض کنید شما مسئولیت توسعه ی یک نمونه اولیه[3] سیستم نرم افزاری را دارید. پس از توسعه آن، مدیر شما از خروجی محصول بسیار تحت تأثیر قرار و پیشنهاد میکند که باید از آن به عنوان یک سیستم اصلی و خروجی نهایی استفاده شود و در صورت لزوم ویژگی های جدیدی اضافه شود. این امر از هزینه توسعه سیستم جلوگیری میکند و میتوان بلافاصله از سیستم ها استفاده کرد. یک گزارش کوتاه برای مدیر خود بنویسید و توضیح دهید که چرا نباید از سیستم های اولیه به عنوان سیستم های اصلی استفاده کرد.

الف)

Prototype:

:این مدل یک مدل چرخه عمر توسعه نرم افزار است و زمانی استفاده می شود که مشتری به طور کامل در مورد محصول نهایی و الزامات آن اطلاع نداشته باشد. بنابراین در این مدل ابتدا یک نمونه اولیه از محصول نهایی توسط توسعه دهندگان توسعه داده می شود، سپس تست شده و طبق بازخورد مش تری تغییراتی ایجاد می شود تا مشتری از نمونه اولیه راضی باشد.

[3] prototype

Spiral:

این مدل نیز یک مدل چرخه عمر توسعه نرم افزار است که برای مدل های ریسک محور بسیار مورد استفاده قرار می گیرد. بر اساس الگوهای ریسک یک پروژه معین، مدل Spiral به توسعه دهندگان کمک می کند تا کارایی مدل را افزایش دهند چرا که بیشتر ریسک ها از قبل مدیریت شده اند؛ در واقع شامل تعدادی حلقه است که یک شکل مارپیچی را تشکیل می دهند که هر حلقه فاز چرخه توسعه نرم افزار نامیده می شود.

Spiral Model	Prototype Model
یک مدل توسعه نرم‌افزار مبتنی بر ریسک است و با ویژگی‌های مدل‌های incremental، آبشاری یا تکاملی ساخته شده است.	یک مدل توسعه نرم‌افزار است که در آن یک نمونه اولیه ساخته شده، تست می‌شود و مطابق با نیاز مشتری پالایش می‌شود.
فازها: 1. Planning Phase 2. Risk Analysis Phase 3. Engineering Phase 4. Evaluation Phase	**فازها:** 1. Requirements 2. Quick Design 3. Build Prototype 4. User Evaluation 5. Refining Prototype 6. Implement and Maintain
تعامل مستمر و پیوسته‌ای با مشتری وجود ندارد	تعامل با مشتری تا زمان تایید prototype وجود دارد
زمانی که نیازهای مشتری مشخص باشد بهترین گزینه است	زمانی که نیاز مشتری مشخص نیست و ممکن است تغییر کند بهترین گزینه است.
بهبود کیفیت می تواند قیمت تمام شده محصول را افزایش دهد	بهبود کیفیت باعث افزایش قیمت تمام شده محصول نمی شود
به تجزیه و تحلیل ریسک توجه ویژه‌ای می کند	بر تحلیل ریسک تاکید نمی کند
پروژه در مقیاس کم تا متوسط حفظ می شود	پروژه در مقیاس بزرگ حفظ می شود
مزایا: • توسعه سریع • توسعه کلیه مراحل به صورت کنترل شده انجام می‌شود • بازخورد مشتری برای تغییرات در صورت نیاز در نظر گرفته می شود • مناسب برای پروژه های با مقیاس بزرگ • مناسب برای پروژه های پرریسک • تجزیه و تحلیل مداوم ریسک که به توسعه بهتر کمک می کند • برای نیازهای به سرعت در حال تغییر مناسب است	**مزایا:** • کاربران نهایی به شدت در کل فرآیند توسعه دخیل هستند • اورها به راحتی شناسایی می‌شوند • تغییرات را میتوان به راحتی انجام داد • بازخورد مداوم کاربر، عملکرد مناسب و موردنظر نمونه اولیه را تضمین می کند • کاربران تصور بهتری در مورد محصول خود دارند
معایب: • برای پروژه های کوچک به دلیل هزینه بالایی که با فرآیند توسعه همراه است، مناسب نیست • نیاز به تیم تحلیل ریسک ذی‌صلاح • وجود احتمال بالای کمبود بودجه یا کمبود وقت (نرسیدن به ددلاین زمانی یا هزینه‌ای)	**معایب:** • تحلیل مسئله ناقص • مشارکت مداوم کاربر ممکن است منجر به افزایش پیچیدگی شود

ب)

همانطور که یک معمار بعد از ساختن یک ماکت مقوایی از خانه اقدام به فروش خانه نمیکند، پس از ساخت پروتوتایپی ک سیستم نیز نمی توان آن را وارد بازار کرد. پروتوتایپ ها وظیفه دارند به ما کمک کنند درک بهتری از محصول داشته باشیم و برخی ویژگی های آن را صحت سنجی کنیم. حقیقت این است که پروتوتایپ ها استحکام یا قابلیت پشتیبانی لازم را برای ورود به پیاده shortcut بازار ندارند چرا که در هنگام ساخت پروتوتایپ بسیاری از ویژگی ها و عملکرد های سیستم صورت سازی میشوند، به این معنا که به صورت خیلی ساده تر و مینیمال تر از عملکرد اصل ی پیاده سازی میشوند و حتی فرآیند دیباگینگ درستی روی آنها صورت نمیگیرد. در بیان کلی تر نمونه های اولیه ممکن است ظاهر مشابهی با محصول نهایی داشته باشند و یا کار کردن با آنها حس مشابهی را داشته باشد ولی در باطن بسیاری از ویژگی و توابع و عملکردهای آنها پیاده سازی نشده اند.در زمان ساخت محصول اصلی ما تلاش میکنیم که سیستم، اصولی، دارای پشتیبانی های کافی و استحکام زیاد باشد ولی در زمان ساخت نمونه اولیه پیاده سازی نامرتب و غیر اصولی و سریع است. بنابراین اگر نمونه اولیه به عنوان محصول نهایی وارد بازار شود عملکرد مطلوبی نخواهد داشت و این عدم عملکرد مناسب میتواند کاربران را نسبت به محصول ما بدبین کند و بازار خود را از دست بدهیم. پس در مجموع ارائه پروتوتایپ به عنوان محصول نهایی نه تنها کاهش هزینه مالی را به همراه ندارد بلکه میتواند به برند و اعتبار محصول/ شرکت نیز آسیب بزند و این مورد جبران پذیر نیست و یا به سختی جبران می شود.

۴. فعالیت های چتری[۴] در طول فرآیند نرم افزار رخ می دهد. آیا فکر می کنید آنها به طور مساوی در سراسر فرآیند اعمال می شوند، یا برخی در یک یا چند چارچوب فعالیت متمرکز شده اند؟ با ذکر مثال توضیح دهید.

به طور کلی، فعالیت های چتر به طور یکنواخت در سراسر فرآیند نرم افزار اعمال نمی شوند. درعوض، بسته به نیازها و اهداف خاص پروژه، فعالیتهای چتر مختلف ممکن است در یک یا چند فعالیت چارچوب متمرکز شوند. به عنوان مثال، فعالیت چتری مد یریت ریسک به ویژه در طول مراحل برنامه ریزی و نیازمندی های فرآیند نرم افزار مهم است، زیرا به شناسایی ریسک ها ی بالقوه و توسعه استراتژی هایی برای کاهش آنها کمک می کند. از سوی دیگر، فعالیت های چتری تضمین کیفیت و آزمایش بیشتر بر مراحل ساخت و آزمایش فرآیند متمرکز است، جایی که به اطمینان حاصل میشود .که نرم افزار استانداردهای کیفی مورد نیاز را برآورده میکند و نقص ها قبل از استقرار شناسایی و رفع میشوند. بنابراین در حالی که فعالیت های چتری در سراسر فرآیند نرم افزار مهم هستند، تاکید و تمرکز آنها ممکن است بسته به فاز خاص فرآیند و نیازهای پروژه متفاوت باشد.

۵. ویژگی های RUP[۵] را ذکر کنید و برای یک فروشگاه اینترنتی هریک از فاز ها و workflow های آن را شرح دهید.

[4] Umbrella Activities
[5] Rational Unified Process

RUP یا Rational Unified Process یک روش توسعه نرم‌افزاری است که شامل چهار فاز اصلی (شناخت، طراحی، پیاده‌سازی و تست) و هر فاز شامل چندین workflow است که به صورت iterative و incremental انجام می‌شود. ویژگی‌های این روش عبارتند از:

۱. Use-case driven: به محور استفاده (use-case) متمرکز است که نیازمندی‌های عملکردی سیستم را از دید کاربرانش ثبت می‌کند.

۲. معماری محور (Architecture-centric) : برمعماری سیستم تأکید قوی دارد و آن را به صورت تکراری و تدریجی در طول پروژه توسعه می‌دهد.

۳. تکراری و تدریجی (Iterative and incremental): یک فرایند تکراری و تدریجی است که به معنای تجزیه پروژه به یک سری از قطعات کوچکتر و قابل مدیریت‌تر است که هر یک نسخه کارآمدی از سیستم را تولید می‌کند.

۴. تمرکز بر مدیریت خطر: برای مدیریت خطرهای پروژه طراحی شده است و شامل فرآیندهایی برای شناسایی، اولویت‌بندی و کاهش خطرات در طول چرخه عمر پروژه می‌شود.

Inception	
Business modeling	پرسجو از کارفرما در رابطه با ماهیت فروشگاه و شناخت بیشتر کار مانند نوع محصولاتی که قرار است در این فروشگاه باشند
Requirements	در آوردن vision و ساخت یکسری از Use case model های جنرال مانند استفاده از درگاه پرداخت
Analysis and Design	تا حد کمی راجب طراحی موجودیت‌ها و دیگر موارد اولیه به صورت خیلی کلی و جنرال
Implementation	کاری انجام نمی‌شود
Test	کاری انجام نمی‌شود
Deployment	کاری انجام نمی‌شود
Configuration Management	کاری انجام نمی‌شود
Management	انتخاب رویکرد توسعه نرم‌افزار به عنوان مثال Agile و برنامه‌ریزی اولیه برای توسعه و درآوردن لیستی از ریسک‌ها و مدیریت آن‌ها
Environment	ایجاد محیط توسعه به عنوان مثال آماده کردن محیط توسعه در شرکت

Elaboration	
Business modeling	ادامه فرآیند مدل‌سازی کار برای تکمیل کردن گپ‌های مرحله قبل
Requirements	تکمیل کردن vision و ساخت و بررسی Use case model های جزئی‌تر مانند نیازمندی‌های کاربران و تکمیل SRS
Analysis and Design	طراحی SAD با بررسی ۴+۱ معماری view model برای طراحی سیستم بر اساس Use case های که مطرح شده‌اند به عنوان مثال برای سبد خرید کاربر چه کامپوننت‌هایی نیاز داریم و این سبد خرید باید به چه صورت ذخیره بشود
Implementation	شروع مرحله اول پیاده‌سازی که شامل پیاده‌سازی معماری پایه‌ای سیستم به عنوان مثال MVC می‌باشد
Test	در کنار توسعه اولین component test ها برای آزمایش کامپوننت‌های ایجاد شده باید نوشته بشوند
Deployment	کاری انجام نمی‌شود
Configuration Management	مدیریت نسخه‌های اجرای اولیه سیستم به عنوان مثال تهیه یک نسخه proto برای ارائه
Management	برگزیدن افراد لازم برای توسعه و آپدیت کردن Risk list و SDP و Business case ها
Environment	آپدیت کردن وضعیت توسعه

Construction	
Business modeling	عملا کاری انجام نمی‌شود
Requirements	عملا کاری انجام نمی‌شود
Analysis and Design	تکمیل مدل‌های آنالیز و طراحی، به عنوان مثال طراحی تمامی معماری نرم‌افزار و تمام کامپوننت‌هایی که نیاز داریم
Implementation	طراحی کامل مدل‌ها و ویژگی‌های که سیستم نیاز دارد، به عنوان مثال در این مرحله باید بتوانیم یک فرآیند افزودن به سبد خرید و ثبت سفارش را انجام بدهیم
Test	نوشتن unit test ها برای آزمایش سیستم و تعامل کامپوننت‌ها با یکدیگر
Deployment	آماده کردن محیط برای اجرای نرم‌افزار به عنوان مثال یک محیط staging برای اولین اجرای نرم‌افزار باید ایجاد کنیم
Configuration Management	مدیریت نسخه‌های جدید ایجاد شده
Management	مدیریت خروجی کار، بررسی مشکلات، تهیه راهنمای استفاده و فرآیند نصب سیستم
Environment	کاری انجام نمی‌شود

Transition	
Business modeling	کاری انجام نمی‌شود
Requirements	کاری انجام نمی‌شود
Analysis and Design	کاری انجام نمی‌شود
Implementation	درست کردن bug هایی که از تست کاربران به دست رسیده است و تعامل آن با دیگر سیستم‌ها، به عنوان مثال اتصال به درگاه پرداخت
Test	انجام تست بتا برای بررسی سیستم و رفع نقص‌های احتمالی
Deployment	ایجاد کردن مستندات برای نرم‌افزار و بررسی مستندات نصب سیستم و اجرای آن و دادن وضعیت از خروجی سیستم
Configuration Management	دادن نسخه‌های نهایی
Management	دریافت نظر از کاربران، بررسی کیفیت کار انجام شده
Environment	کاری انجام نمی‌شود

۶. مزایای اصلی فرآیند RUP چیست، به نظر شما در چه سازمان ها و سیستم هایی بکار می‌رود و چرا آن را یکپارچه می نامند؟

ریسک ها را به آسانی برطرف میکند :اطمینان حاصل میکند که مشتریها در راستای موارد مورد نیاز خود قرار بگیرند و همچنین، منابع کمتری برای ادغام فرایندها در توسعه نرم افزارها استفاده میشود .تغییرات را کنترل میکند :این فرآیند، هماهنگ سازی اجزای مختلف پروژه که توسط تیمهای مختلف انجام میشود را آسانتر میکند .در برگیرنده الگوهای انعطاف پذیری است :توسعه نرما فزارها به مدیران این فرصت را می دهد تا از فرآیندهای رسیدگی به مشکلات رایج، مجدد استفاده کنند. از آنجایی که پروژه ها مشابه هم نیستند، امکان تغییر فرآیندهای خاص به منظور رسیدگی به نیازهای پروژه، راه حلی کارآمد خواهد بود .پروژه کارآمدتری را تحویل میدهد RUP :. منابع مورد نیاز کل پروژه را در اختیار مدیران میگذارد و بهترین شیوههایی که در جهان پذیرفته شده است را در طول پروژه استفاده میکند .از توسعه تکرار شونده حمایت میکند :سیستم توسعه نرمافزار RUP ، در مرحلهای است که اطمینان میدهد در هر فرآیند، تکرارها قابل اجرا هستند .بطور کلی فرآیند RUP بهره وری تیم را با فراهم نمودن دسترسی تمام افراد تیم به یک پایگاه دانش سهل الوصول به همراه راهنماها، الگوها و ابزارهای کمکی برای همه فعالیت های بحرانی توسعه، افزایش میدهد . فرآیند RUP، برای سازمانهای بزرگ توسعه نرم افزار مناسب تر است اما برای تیم های کوچک نیز می تواند مفید باشد. فرایند RUP یک فرایند توسعه نرمافزار کامل است و بنابراین می تواند برای توسعه هر محصولی اعم از کوچک و بزرگ استفاده شود .اما به دلیل دارا بودن پروسه های زمان بر و سنگین برای پروژه های کوچک استفاده نمیشود. بلکه پروژه های بزرگی که پیچیدگی دارند و یا نیازمندی های آن تغییر می کند مناسب است.

این متدولوژی از یکپارچگی سه متدولوژی معروف دیگر به وجود آمده است که شامل OMT OSE , booch می شود. از UML در جهت کار خود استفاده می کند و در واقع می توان گفت UML خود ثمره RUP است و این یک ویژگی منحصر به فرد است که متدولوژی با خودش گسترش می یابد. مفاهیمی از قبیل object , class و ... مفاهیم ساده و ثابتی هستند ولی متدولوژی های قبلی شاخصه های خاصی داشتند که اکنون همه ی آن ها یکسان شده اند.

۷. با استفاده از قالب مستندات RUP برای یک سامانه‌ی درگاه پرداخت اینترنتی نیازمندی های سیستم را توصیف کنید و plan مدیریت پروژه را نیز با استفاده RUP تهیه کنید. توصیف نیازمندی های شما حداقل باید شامل use-case model و SRS باشد(برای این قسمت شرح یکی از use-case با استفاده از توصیف use-case در RUP بنویسید) و فرآورده های نظم مدیریت پروژه شما باید حداقل شامل Risk list و طرح توسعه ی نرم افزار باشد.

یک نمونه ی درست :

Risk list

Risk List

1. Introduction

The introduction of the provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this

1.1 Purpose

Specify the purpose of this

1.2 Scope

A brief description of the scope of this; what Project(s) it is associated with and anything else that is affected or influenced by this document.

1.3 Definitions, Acronyms, and Abbreviations

This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the. This information may be provided by reference to the project's Glossary.

1.4 References

*This subsection provides a complete list of all documents referenced elsewhere in the **Risk List**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.*

1.5 Overview

*This subsection describes what the rest of the **Risk List** contains and explains how the document is organized.*

2. Risks

2.1.1 Risk Magnitude or Ranking

An indicator of the magnitude of the risk may be assigned to help rank the risks from the most to the least damaging to the project.

2.1.2 Description

A brief description of the risk.

2.1.3 Impacts

List the impacts on the project or product.

2.1.4 Indicators

Describe how to monitor and detect that the risk has occurred or is about to occur. Include such things as metrics and thresholds, test results, specific events, and so on.

2.1.5 Mitigation Strategy

Describe what is currently done on the project to reduce the impact of the risk.

2.1.6 Contingency Plan

Describe what the course of action will be if the risk does materialize: alternate solution, reduction in functionality, and so forth.

2.2 <next Risk Identifier—a descriptive name or number>

Table of Contents

Revision History

Date	Version	Description	Author
02/04/2023	2.5	refactoring	Pouria

software development plan

Revision History

Date	Version	Description	Author
02/04/2023	2.5	refactoring	Pouria

Table of Contents

Software Development Plan

1. Introduction

*[The introduction of the **Software Development Plan** should provide an overview of the entire document. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Software Development Plan**.]*

1.1 Purpose

*[Specify the purpose of this **Software Development Plan.**]*

1.2 Scope

*[A brief description of the scope of this **Software Development Plan**; what Project(s) it is associated with, and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms and Abbreviations

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Development Plan**. This information may be provided by reference to the project Glossary.]*

1.4 References

*[This subsection should provide a complete list of all documents referenced elsewhere in the **Software Development Plan**. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.*

*For the **Software Development Plan**, the list of referenced artifacts should include:*

- *Iteration Plans*
- *Requirements Management Plan*
- *Measurement Plan*
- *Risk Management Plan*
- *Development Case*
- *Business Modeling Guidelines*
- *User Interface Guidelines*
- *Use-Case-Modeling Guidelines*
- *Design Guidelines*
- *Programming Guidelines*
- *Test Guidelines*
- *Manual Style Guide*
- *Infrastructure Plan*
- *Product Acceptance Plan*
- *Configuration Management Plan*
- *Evaluation Plan (only if this is a separate plan - normally this is part of the SDP at Section 6.2)*
- *Documentation Plan*

- *Quality Assurance Plan*
- *Problem Resolution Plan*
- *Subcontractor Management Plan*
- *Process Improvement Plan]*

1.5 Overview

*[This subsection should describe what the rest of the **Software Development Plan** contains and explain how the document is organized.]*

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

[A brief description of the purpose and objectives of this project and a brief description of what deliverables the project is expected to deliver.]

2.2 Assumptions and Constraints

[A list of assumptions that this plan is based and any constraints, for example. budget, staff, equipment, schedule, that apply to the project.]

2.3 Project Deliverables

[A tabular list of the artifacts to be created during the project, including target delivery dates.]

2.4 Evolution of the Software Development Plan

*[A table of proposed versions of the **Software Development Plan**, and the criteria for the unscheduled revision and reissue of this plan.]*

3. Project Organization

3.1 Organizational Structure

[Describe the organizational structure of the project team, including management and other review authorities.]

3.2 External Interfaces

[Describe how the project interfaces with external groups. For each external group, identify the internal and external contact names.]

3.3 Roles and Responsibilities

[Identify the project organizational units that will be responsible for each of the disciplines, workflow details, and supporting processes.]

4. Management Process

4.1 Project Estimates

[Provide the estimated cost and schedule for the project, as well as the basis for those estimates, and the points and circumstances in the project when re-estimation will occur.]

4.2 Project Plan

4.2.1 Phase Plan

[Include the following:

- *Work Breakdown Structure (WBS)*
- *a timeline or Gantt chart showing the allocation of time to the project phases or iterations*
- *identify major milestones with their achievement criteria*

Define any important release points and demos]

Electronic Payment System	Version: 2.5
Software Requirements Specification	Date: 22/01/2023
E345	

Revision History

Date	Version	Description	Author
21/04/2023	2.5	Refactoring	Pouria

Software Requirements Specification

Software Requirements Specification

1. Introduction

In this SRS document, all details relevant to Transaction Processing will be analyzed.

To be more specific, requirements related to integrity, security and confidential will be elaborated.

1.1 Purpose

The purpose of this requirement is to ensure safe transactions for the online shopping store.

1.2 Scope

This feature will be applied to all parts of the system because the main responsibility of a shopping website is to ensure safe and correct transition processes

1.3 Definitions, Acronyms, and Abbreviations

JWT => Jason web token

RB => role back

T => transaction

OTP => one time password

1.4 References

E123 => SRS for reliability

F645 => SRS for authentication

1.5 Overview

In the rest of document, transaction processing will be elaborated.

2. Overall Description

Introduction

Body

conclusion

2.1 Use-Case Model Survey

In this subsystem, after that a client buy a good, he takes actions to pay for the price and this feature is responsible for handling transactions

2.2 Assumptions and Dependencies

This feature is highly dependable on Banks' databases for transactions

3. Specific Requirements

Atomic transactions

Authentication

3.1 Use-Case Reports

A client will enter his card information and after that he applies for paying the product and this feature should provide safe transactions for him.

3.2 Supplementary Requirements

API of banks

:Use-Case Specification

Use-Case Specification: Transaction Processing>

1. Use-Case Name

1.1 Brief Description

[The description briefly conveys the role and purpose of the use case. A single paragraph will suffice for this description.]

2. Flow of Events

2.1 Basic Flow

. A new user is created in the system

The user created is granted the privileges as specified

The user created is created with a unique 4 digit employee number and a default password

2.2 Alternative Flows

2.2.1 Failure

- The system administrator is presented with an error message

3. Special Requirements

Ccv2 and otp password should be entered

4. Preconditions

The user must be logged-in as "System Administrator"

5. Post conditions

The user created is granted the privileges as specified

6. Extension Points

پایان