الغاية من الجلسة: التعرف على وثيقة توثيق المتطلبات SRS في تطوير البرمجيات.

مقدمة:

وثيقة متطلبات البرمجيات (SRS) هي وصف مفصل لنظام برمجي يجب تطويره مع جميع متطلباته الوظيفية وغير الوظيفية. تتم إعداد وثيقة متطلبات البرمجيات بناءً على الاتفاق بين العميل والمطورين. يمكن أن تتضمن SRS حالات الاستخدام التي تشرح كيفية تفاعل المستخدم مع نظام البرمجيات. تتكون وثيقة متطلبات البرمجيات من جميع المتطلبات اللازمة لتطوير المشروع. لتطوير نظام البرمجيات، يجب أن يكون لدينا فهم واضح للنظام البرمجي، ولتحقيق ذلك يجب أن نحافظ على تواصل مستمر مع العملاء لجمع جميع المتطلبات.

1. تعريف وثيقة المتطلبات (SRS):

وثيقة المتطلبات هي وثيقة تصف بدقة ما يجب أن يقوم النظام بفعله. تحتوي على تفاصيل حول المتطلبات الوظيفية وغير الوظيفية للنظام المراد تطويره.

2. أهمية وثيقة المتطلبات:

- تحديد المتطلبات: تساعد SRS في تحديد وتوثيق جميع المتطلبات التي يجب أن يلبيها النظام.
- توضيح الفهم المشترك: تعمل كوسيلة لتحقيق فهم مشترك بين العملاء وفرق التطوير بشأن ما يجب أن يحقق النظام.
- توجيه عملية التصميم: تمثل SRS دليلًا لفرق التصميم لبناء النظام وفقًا للاحتياجات المحددة.
- أساس للتقييم: توفر وثيقة المتطلبات أساسًا لتقييم الأداء اللاحق للنظام والتحقق من أنه يلبي توقعات العملاء.

3. مكونات وثيقة المتطلبات:

- وصف المنتج: توفير نظرة عامة عن النظام والغرض من تطويره.
- متطلبات النظام: تفاصيل حول المتطلبات الوظيفية وغير الوظيفية للنظام.
- واجهة المستخدم: وصف لواجهة المستخدم وتفاصيل تصميمها.
- قيود المشروع: تحديد أي قيود تقنية أو زمنية قد تؤثر على تصميم وتطوير النظام.

4. عملية إعداد وثيقة المتطلبات:

- جمع المتطلبات: التفاعل مع العملاء والمستخدمين لفهم احتياجاتهم ومتطلباتهم.
- توثيق المتطلبات: تحديد وتوثيق جميع المتطلبات بشكل واضح ودقيق.
- التحقق من الصحة: التحقق من أن جميع المتطلبات قابلة للقياس ومتناسبة مع أهداف النظام.

5. تحديث وثيقة المتطلبات:

يجب تحديث SRS في حالة حدوث أي تغييرات في المتطلبات أثناء تطور المشروع.

6. أمثلة لأدوار تأليف وثيقة المتطلبات:

- محلل المتطلبات: يتولى جمع وتوثيق المتطلبات.
- مدير المشروع: يضمن أن تكون المتطلبات قابلة للتحقق وفي متناول اليد.
- فريق التطوير: يعتمد على المتطلبات لتصميم وتطوير النظام.

باختصار، وثيقة المتطلبات هي أحد المكونات الأساسية في عملية تطوير البرمجيات، حيث تسهم في تحقيق فهم واضح ومشترك لجميع الأطراف المعنية وتحديد الاتجاه الصحيح لتصميم وتنفيذ النظام.

صفات الـ SRS الجيدة:

1. الصحة (Correctness):

الصحة تعني أن متطلبات الوثيقة صحيحة ودقيقة.

2. الاكتمال (Completeness):

تعتبر SRS كاملة إذا تضمنت العناصر التالية:

أ- جميع الاحتياجات الأساسية:

سواء كانت تتعلق بالوظائف أو الأداء أو التصميم أو القيود أو السمات أو الواجهات الخارجية.

ب- تحديد استجابات البرنامج لجميع فئات البيانات الممكنة:

ضمن كافة فئات الحالات الممكنة، يجب تحديد كيف سيتفاعل البرنامج مع جميع أصناف البيانات الممكنة.

ملحوظة: من المهم تحديد استجابات البرنامج للقيم الصحيحة والقيم غير الصحيحة.

ت- وسوم كاملة وإشارات إلى جميع الشكلين والجداول والرسومات:

يجب أن تكون هناك تسميات كاملة وإشارات إلى جميع الأرقام والجداول والرسوم التوضيحية في SRS، بالإضافة إلى تعريفات لجميع المصطلحات ووحدات القياس.

3. التناسق Consistency:

بمعنى أن المتطلبات يجب أن تكون متناسقة مع بعضها البعض وليس العكس، فعلى سبيل المثال من الخطأ أن نضع في متطلب ما أن الأضواء جميعها خضراء ونعود نكتب الأضواء صفراء في مكان آخر، أو أن نكتب أن التقارير يجب أن تكون على شكل جداول وفي مكان آخر من الوثيقة تكون التقارير على شكل ملف نصي.

4. البعد عن الغموض Unambiguousness:

أي يجب أن يكون كل شيء واضحاً في وثيقة المتطلبات من رموز وتعاريف وأشكال.

5. التصنيف حسب الأهمية والاستقرار Ranking:

تصنيف وثيقة متطلبات البرمجيات حسب الأهمية والاستقرار يعني أن كل متطلب فيها يحمل معرِّفاً للدلالة على أهمية أو استقرار تلك المتطلبات بشكل خاص.

عادةً، لا تكون جميع المتطلبات مهمة بنفس الدرجة. قد تكون بعض المتطلبات أساسية، خاصةً في تطبيقات تتعلق بالحياة، في حين يمكن أن تكون أخرى مرغوبة فقط. يجب تحديد كل عنصر لجعل هذه الفروق واضحة وصريحة.

من الممكن أن يكون هناك طرق متعددة لتحديد مستوى الأهمية أو الاستقرار، وفي هذا السياق، يُستخدم معرف لكل متطلب للدلالة على هذه المعلومات. يمكن أن يكون ذلك على شكل رقم أو رمز يُرفق بكل متطلب.

على سبيل المثال:

"1" يشير إلى متطلب هام جدًا.

"2" يشير إلى متطلب مهم.

"3" يشير إلى متطلب ذو أهمية متوسطة.

وهكذا...

هذا التصنيف يُساعد على تحديد أولويات التطوير والاختبار، حيث يُمكن للفريق التركيز على تنفيذ المتطلبات الأكثر أهمية أو الأكثر استقرارًا أولاً، مما يسهم في تحقيق نتائج أفضل وفقًا للأولويات المحددة.

6. قابلية التعديل (Modifiability):

يجب أن تكون وثيقة متطلبات البرمجيات قابلة للتعديل بقدر الإمكان وينبغي أن تكون التعديلات مفهومة تماماً.

7. التحقق (Verifiability):

تعتبر وثيقة متطلبات البرمجيات صحيحة عندما يمكن التحقق من أن المتطلبات المحددة يمكن التحقق منها باستخدام نظام فعّال منخفض التكلفة للتحقق مما إذا كان البرنامج النهائي يفي بتلك المتطلبات. يتم التحقق من المتطلبات بمساعدة عمليات المراجعة.

8. التتبع (Traceability):

تعتبر وثيقة متطلبات البرمجيات قابلة للتتبع إذا كان أصل كل متطلب واضحاً وإذا سهلت عملية الرجوع إلى كل شرط في وثائق التطوير أو التحسين المستقبلية.

9. قابلية الفهم من قبل العميل Understandable:

يجب أن تكون وثيقة متطلبات البرمجيات قابلة للفهم من قبل العميل، حيث يمكن أن يكون المستخدم النهائي خبيراً في مجاله الخاص، ولكن قد لا يكون مدرباً في علوم الحاسوب. لذا، يجب تجنب الاعتماد على الرموز والتعابير الرسمية إلى أقصى حد ممكن. يجب الحرص على أن يكون لغة الوثيقة بسيطة وواضحة.

في سبيل تحقيق قابلية الفهم للعميل:

تجنب الرموز المعقدة: تجنب استخدام رموز معقدة أو لغة فنية صعبة الفهم قدر الإمكان، بحيث يمكن للعميل العادي فهم مضمون الوثيقة.

استخدام لغة بسيطة: يجب أن تكون اللغة مبسطة وواضحة، مع تجنب استخدام مصطلحات فنية معقدة قد تكون غير مألوفة للعميل.

توضيح الأفكار: يمكن توضيح الأفكار باستخدام أمثلة وشرح إضافي لضمان تفهم دقيق للمتطلبات.

تحديد الأهداف الرئيسية: يجب أن يتم تحديد الأهداف والمتطلبات الرئيسية بوضوح، مما يسهل على العميل التركيز على النقاط الرئيسية.

إن القالب الأشهر والمعتمد هو IEEE 1998 وهو القالب الـ standard في مجال توثيق المتطلبات.

إن القالب موجود على الموقع، بالإضافة إلى ملفين كأمثلة عليه.