SOFTWARE TEST DOCUMENT

Home Automation System

Nickolas Helgeland, Jon Erik Nordskog og Kristian Sande Sjølyst

Innhold

1 Introduksjon

Home Automation System er et system som registrerer verdier fra sensorer installert på en eller flere eiendommer. Temperatursensor, bevegelsessensor og fuktsensor er eksempler på sensorer som er planlagt til å kunne brukes i systemet. Frem til RTM-utgivelsen bruker vi kun temperatursensorer, men systemet er laget for å kunne utvides med sensortypene som er nevnt i tillegg hvilke som helst andre sensorer. Det er også lagt inn mulighet for å styre ting i hjemmet som lys, varme og sikkerhetssystemer som alarm. Dette er ikke tenkt utviklet i dette stadiet og kommer som funksjonalitet i senere versjoner.

Testingen vil skje i to nivåer:

1. **Nivå 1** der det kreves lite kunnskap om programvare og elektronikk. Her testes utseende og funksjonalitet på GUI. Altså UX eller user experience.
2. **Nivå 2** der det kreves god kunnskap om database og om transdusere. Kunnskap om transdusere, for å kunne sjekke om det digitale signalet er en riktig representasjon om den faktiske fysiske verdien. Være seg f.eks. temperatur eller fukt.

2 Systemoppdeling

Systemet er delt opp i tre deler som vi fordeler på to testnivåer.

Figur 2.1 Systemskisse

2.1 WebApp, **nivå 1**

Programvare som leser fra databasen, kalt **WebApp**, viser data på websiden. Fra websiden skal det også kunne gjøres endringer i innstillinger og brukeren. Denne delen testes i nivå 1.

2.2 SensorGetApp, **nivå 2**

Programvaren som er installert i en boks, kalt **SensorGetApp**, leser fra sensorene hos kunden og sender data til databasen:
Her må det bl.a. testes om dataene blir lagret på rett måte og at programmet henter innstillingene fra databasen.

2.3 Database, **nivå 2**

SQL databasen, kalt **databasen**. Her er last- og stresstest sentralt. Tåler databasen utvidelser og høy belastning med mange samtidige brukere og mange sensorer?

2.4 Mål for testen

Testens mål er å finne bugs som gir dårlig brukere en dårlig opplevelse eller viser feil informasjon

1. At systemet fungerer ihht Software Development Plan.
2. At programvaren i boksen fungerer uten at kunden trenger å gjøre noe.
3. At kunden kan logge inn og at GUI er enkelt og intuitivt.
4. At kunden kan logge inn i tjenesten og se sensorverdier.

2.5 Begrensninger

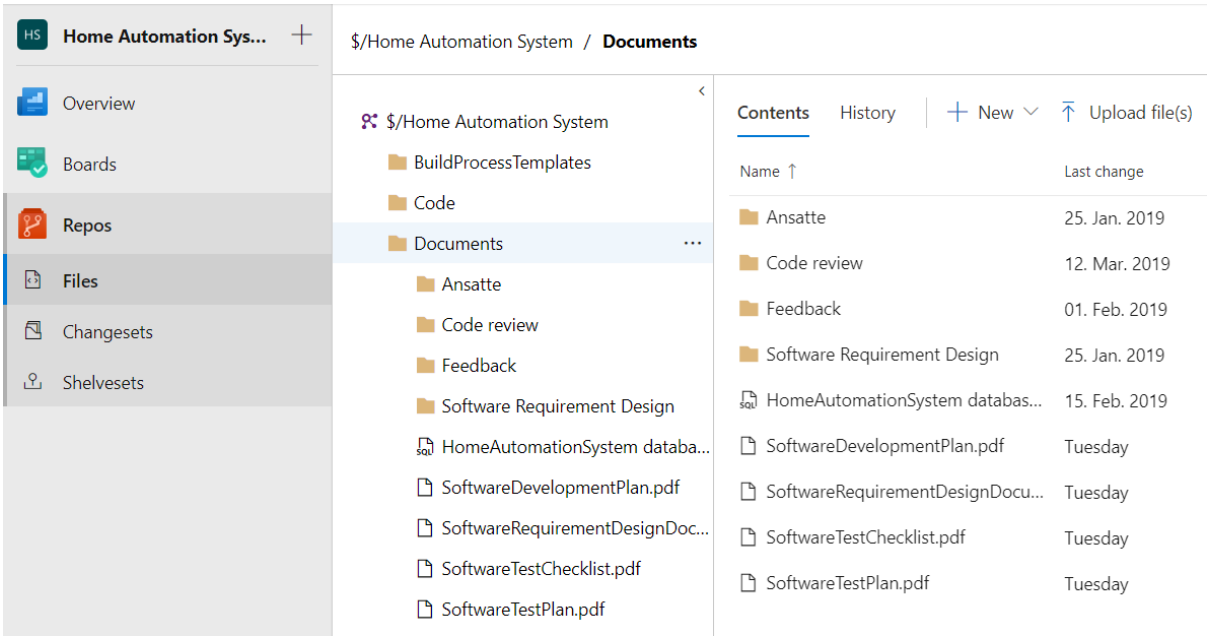
Noen ubrukte elementer vises i GUI, men er forbeholdt fremtidige utgivelser. Dette blir diskutert senere i dokumentet. Disse er ikke en del av testregimet.

2.6 Andre dokumenter for referanser

Følgende dokumenter er utarbeidet som lager grunnlaget for STP.

1. Software Developmet Plan Document, fra nå kalt **SDP**. SDP finner en i Azure DevOps under repost/Documents. Se Bilde 2.1.

2. Software Reqirement & Design Document, fra nå kalt **SRD**. SRD finner en i Azure DevOps under repos/Documents. Se Bilde 2.1.



Bilde 2.1 Filbanen til dokumentene i Azure DevOps

3 Om testmiljøet

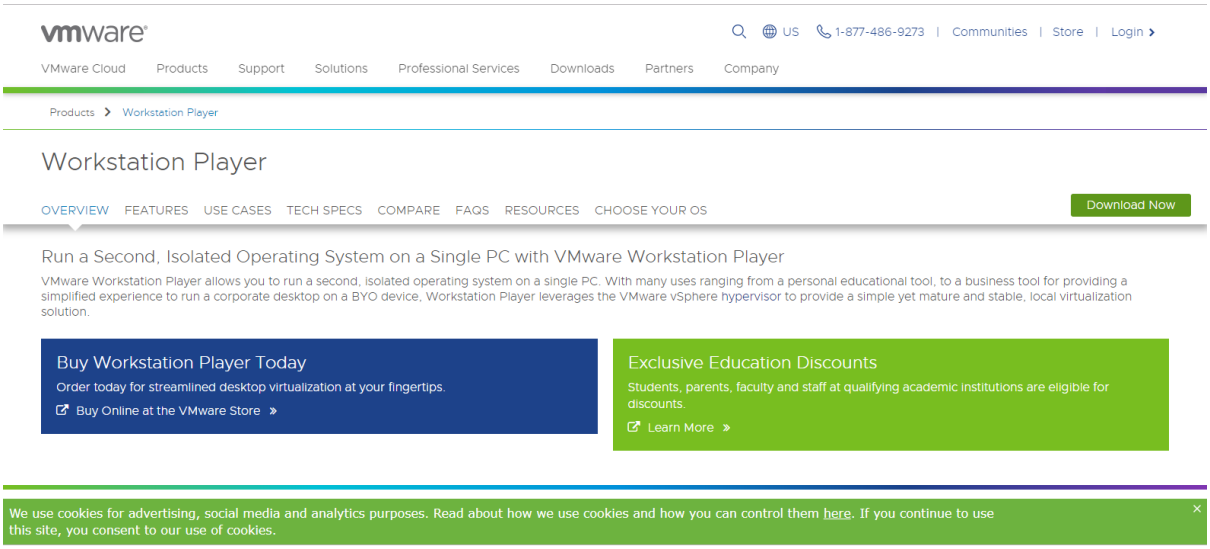
Testerne skal teste systemet i et virtuelt miljø. Det er laget en virtuell maskin som kan kjøres i VMWare Workstation Player. Her er alt det nødvendige allerede installert:

1. Windows 10 Edu edition.
2. VMWare Tools.
3. SQL Server Express 2017.
4. SQL Server Management.
5. NI-drivere.
6. SQL-script er kjørt for å opprette databasen, views og stored procedures.
7. Desktop app.
8. Wep-app.

3.1 Trinn for trinn-oppsett av virtuelt miljø

For disse trinnene for å komme i gang med testmiljøet:

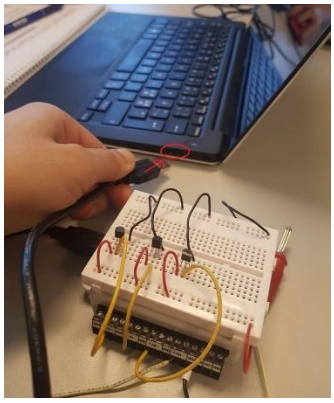
1. Laste ned og installer VMWare Workstation Player. Denne er gratis og lastes ned herfra: https://www.vmware.com/products/workstation-player.html



Figur 3.1Nedlasting av VMWare Workstation Player

2. Kontakt din utvikler ut utdeling av virtuell maskin på minnepinnne.
3. Starte virtuell maskin i VMWare Workstation Player.
4. Når Windows har startet plugger du inn NI-Daq til USB. Se Bilde 3.1 Tilkobling av NI-Daq til USB.Se Bilde 3.1. Enheter som kobles til maskinen etter at den virtuelle maskinen har startet

vil automatisk routes til det virtuelle miljøet.



Bilde 3.1 Tilkobling av NI-Daq til USB.

5. Du er nå klar til å starte testingen ihht SoftwareTestChecklist.xlsx på det nivået du har blitt satt til å teste.

4 Testpersonell og ansvars

Testingen utføres parallelt med av to eller tre personer der Nickolas Helgeland har ansvar som testleder. Det er forventet at testlederen har mer inngående kunnskap og erfaring fra testing enn resten av testteamet. Testlederen har ansvar for at testene blir korrekt utført og at testskjemaene blir sendt til Scrummaster, Jon Erik Nordskog, umiddelbart etter fullført testing.

4.1 Rapportering av bugs

Når testeren oppdager bugs så skal dette rapporteres i Azure DevOps. Dette skal gjøres i disse stegene.

Finn nok informasjon om bugen slik at teamet kan forstå innvirkningen det har på produktet.

Er bugen fixet? Hvordan ble bugen fixet?

Bestem prioriteten av bugen fra høyest til lavest.

Kritisk: Må fikse. En feil som forårsaker oppsigelse av en eller flere systemkomponenter eller hele systemet, eller forårsaker omfattende data korrupsjon. Og det finnes ingen akseptable alternative metoder for å oppnå nødvendige resultater.

High, Vurder å fikse. En feil som forårsaker oppsigelse av en eller flere systemkomponenter eller hele systemet, eller forårsaker omfattende data korrupsjon. Imidlertid eksisterer en akseptabel alternativ metode for å oppnå nødvendige resultater.

Medium, (Standard) En feil som forårsaker at systemet produserer feil, ufullstendige eller inkonsekvente resultater.

Low: En mindre eller kosmetisk feil som har akseptable løsninger for å oppnå nødvendige resultater.

Legg bugen inn i Azure DevOps

Videoguide finner du her: https://youtu.be/4xjo7Bnc5UA

5 Hva skal testes?

Alle fysiske funksjoner og alt i GUI skal testes. Utfør testen ihht SoftwareTestChecklist.xlsx og lett til eventuelle egne tester om det mangler.

5.1 WebApp

All funksjonalitet ihht «Software Test checklist.xlsx». Se fane med navn “Level 1».

5.2 SensorGetApp

All funksjonalitet ihht «Software Test checklist.xlsx». Se fane med navn “Level 2».

5.3 Databasen

All funksjonalitet ihht «Software Test checklist.xlsx». Se fane med navn “Level 2».

6 Oversikt over ulike test typer

50% av tiden som brukes i programvareutvikling forventes brukt til testing. Testingen kan utføres av forskjellig personell med ulik utdanning og erfaring. Her er en oversikt over test typer og testnivåer enkelt forklart.

6.1 Valideringstesting og feiltesting

Under valideringstestingen testes det om systemet fungerer ihht de dokumenterte kravene. Under feiltestingen kjører man en serie med ulike verdier inn til programmet for å se om programmet oppfører seg innenfor kravene og spesifikasjonene som er stilt i dokumentasjonen tidligere. Om dette ikke oppfører seg forventet er dette bugs i programvaren.

6.1.1 Valideringstesting

Under valideringstestingen ser vi etter om systemet som er utviklet tilfredsstiller Software Requirement and Design-dokumentet.

6.1.2 Feiltesting

Ved feiltesting sjekker vi nøyere hva rapporterte feil består av og hvordan vi best kan rette disse.

6.2 Test nivåer

Testing kan deles inn i flere ulike nivåer basert på om små deler eller større deler av systemet som skal testes.

6.2.1 Unittest

Unit tester blir laget av utviklerene før, under eller etter programmeringen. Testen kjøres separat og tester klasser og metoder.

6.2.2 Regression testing

Her testes om nye komponenter i koden henger sammen med annen og tidligere utviklet kode. Dette kan testes manuelt eller å kjøre unittestene en gang til.

6.2.3 Integration testing

Her testes om alt fungerer sammen. I vårt tilfelle har vi to apper som må fungere sammen.

6.2.4 Systemtesting

Typisk brukes «black-box-testing» for å sjekke om systemet møter de ønskede krav.

6.2.5 Acceptance testing

Her tester kunden systemet for å se om det utviklede systemet kan godkjennes.

7 Hvordan skal det testes?

7.1 WebApp

Sjekk og full ut «Software Test checklist.xlsx». Se fane med navn “Level 1».

7.2 SensorGetApp

Sjekk og full ut «Software Test checklist.xlsx». Se fane med navn “Level 2».

7.3 Databasen

Sjekk og full ut «Software Test checklist.xlsx». Se fane med navn “Level 2».

8 Pass/Fail av kriterier

For at testene skal være godkjent så må minst 80% av funksjonaliteten som er tilgjengelig i programmet fungere. En del funksjonalitet kommer i senere iterasjoner. Om testen ikke er godkjent så skal feilene rettes og en ny test gjøres.

Utfylt og fullført Excel fil sendes til Scrum-master som gjennomgår og arkiverer denne.

9 Tidsplan

Testen skal gjennomføres hver uke og alle feil bør være rettet innen neste utgivelse.

Tabell 9-1 Tidsplan for når testing skal utføres

Utgivelse	**Ukenrummer**	**Ukedag**	**Testnivå**
Beta	Uke 15	Tirsdag	Test på nivå 1
Beta	Uke 15	Fredag	Test på nivå 2
Beta	Uke 16	Tirsdag	Test på nivå 1
Beta	Uke 16	Fredag	Test på nivå 2
Beta	Uke 17	Tirsdag	Test på nivå 1
RC	Uke 17	Fredag	Test på nivå 2
RC	Uke 18	Tirsdag	Test på nivå 1
RTM	Uke 18	Fredag	Test på nivå 2

10 Krav til testerne

Det er ulike krav til dem som skal teste om det skal testes på nivå 1 eller nivå 2.

Generellt kan kreves det at testerne har:

- generell kunnskap om Windows og web.
- tilgang til en ordinær pc med Windows med USB 2.0.
- lest dette dokumentet.
- gjort seg kjent med dokumentene SDP og SRD
- fulgt guiden om nedlastning og installering i kapitel 0.

10.1 Testere på nivå 1

Her kreves det ikke noe spesielt utover den generelle kunnskapen som kreves i forrige kapittel.

10.2 Testere på nivå 2

Testere på nivå 2 må ha god kunnskap om database. Hvordan stored procedures fungerer og kunne bruke queries i SQL Server Management for å lese og analysere data i databasen. I tillegg er det en fordel om testerene på dette nivået kan litt ingeniørteknisk om sensorer og transdusere og måling av dette.

11 Test log

Her er en log over alle testresultatene som er blitt gjort i testene.

Number	Refactored	Description	File, line num...
1		Are all variables written in camelCase?	Program.cs, , ok.
2		Are all class names written in PascalCase?	Program.cs, , ok.
			SensorInn.cd, 16, Is Daq a variable? Should use camelCase.
3		Are all method names names written in PascalCase?	Program.cs, , ok.
4		Are all constant names written in captital letters only?	Program.cs, , ok.
5		Are prefixes for components according to nameconven	Program.cs, , NA.
6		Are all comments relevant to the code?	Program.cs, , All comments must be in english according to SDP.
			Sensor.Inn.cs, 41, Comment does not say it converts to celcius.
			TimerClass.cs, 26-30, Comments are in english, nice!
7		Does large segments of code excist without comments?	Program.cs, , ok.
			Sensor.Inn.cs, 13-22, Needs commenting.
8		Should part of the code be extracted into a method?	Program.cs, , ok.
9		Should part of the code be made into a class?	Program.cs, , ok.
10		Does the code repeate so that a method should med cr	Program.cs, , ok.
			SensorInn.cs, 30-36, Code is duplicated; Line 35-35 should be put outside look.
11		Is there a security issue with the code (Public\private)?	Program.cs, , unsure.
12		Is there a security issue with the code (exteral threat)?	Program.cs, , unsure.
13		Is the code easy to understand?	Program.cs, , ok.
			TimerClass.cs, 36-37, Method inside method inside method inside .
14		Is the code easy to maintain?	Program.cs, , ok.
			Sensor.Inn.cs, 41, Use different method name.
			Sensor.Inn.cs, 43-45, Temp should have celcius in name?
15		Is the code scalale to large amount of data?	Program.cs, , ok.
			Sensor.Inn.cs, 29-36, Duplicate code.
16		Is the code scalale to huge number of users?	Program.cs, , ok.
17		Can enum be used instead of open lists?	Program.cs, , NA.
18		Should another datatype be used? Int\float, array\list	Program.cs, , ok.
19		Should exception handling be implemented (try\catch)?	Program.cs, , Exception should be used so that the program continous and do not crash.
			And throughout the program. Exceptions should be loged and shown in the GUI.
20		Should a stored procedures be created?	Program.cs, , NA.
21		Is the connection to SQL closed?	Program.cs, , NA.
22		Can the code be replaced with framework features?	Program.cs, , ok.