Laboration 3 – testning

Användarfall - Domare
Beskriver nedan användarflödet för en användare med behörigheten domare. För ytterligare beskrivning se användarfall i Laboration 2.

Domare använder systemet för att poängsätta gymnaster under tävling.
Primärflöde

1. Användaren trycker på knappen logga in.
2. Systemet öppnar login sida
3. Användaren skriver in användarnamn och lösenord
4. Användaren trycker på knappen OK
5. Användaren kommer in i systemet och kan använda de funktioner som systemet tillhandahåller.
6. Användaren väljer tävling.
7. Användaren väljer gren.
8. Användaren godkänner deltagarna.
9. Användaren skriver in gymnasternas poäng.
10. Användaren vill avsluta tävlingen.
11. Användaren bekräftar resultaten.

Sekundärflöde 1 – Felaktigt användarnamn

3.A - Användaren skriver in fel användarnamn
4.A - Användaren klickar log in
5.A - Felmeddelande visas och användaren ges ett nytt försök.

Sekundärflöde 2 – Felaktigt lösenord

3.B - Användaren skriver in rätt användarnamn, men fel lösenord
4.B - Användaren klickar login
5.B - Felmeddelande visas och användaren ges ett nytt försök.
3B2 – Efter tre felaktiga försök låses inloggningsmöjligheten för användarnamnet.

Sekundärflöde 3 – Användare har fel behörighet.

5.C – Användaren kommer inte åt nödvändiga funktioner.
5.C.B – Måste kontakta administratör.

Sekundärflöde 4 – Godkänner inte deltagarna

8.A – Användaren tar bort deltagare.
8.A.B – Godkänner deltagare

Sekundärflöde 5 – Användaren skriver in ogiltig poäng

9.A – Får möjlighet att skriva om poäng.
9.A.B – Poängen sparas och användaren kan skriva in poäng för nästa deltagare.

Sekundärflöde 6 – Användaren vill ändra före bekräftelse

10.A – Användaren kommer tillbaka till poängsättning
10.A.B – Får åter möjlighet att bekräfta tävlingen avslutad.

Testplan

Syftet
Testplanen skapar riktlinjer för vilka aspekter av användarfallet som ska testas och hur buggar ska åtgärdas. Planen är främst avsedd för projektledare, utvecklare och testare, men kan i vissa fall även visas för kund i syfte att fastställa önskat resultat.

Testfokus
Systemet kommer att ha många olika typer av användare. Användare med väldigt olika nivå av datorkunskap. Testerna ska i detta skede främst säkerställa användarvänlighet och minska risken för fel grundat den mänskliga faktorn. Systemet ska vara självförklarande och stor vikt ska läggas på att testerna säkerställer att inget användaren matar in kan bli fel och att vid eventuellt felaktiga inmatningar ska användaren vägledas att göra rätt.

Testprocedur
Enhetstester – direkt efter minsta sammanhörande kod är färdigställd. Sker löpande innan commit till Github. Buggar åtgärdas direkt av utvecklare. Saknas möjlighet att åtgärda buggar direkt ska buggkommentar skrivas i samband med commit till github. Eventuella buggar ska alltid dokumenteras under respektive testfall och markeras antingen som åtgärdad eller återstående. Efter slutfört test med önskat resultat ska testfallet markeras och signeras som slutfört.

Integrationstester – direkt när minst två moduler som ska samverka är färdigställda. Sker löpande innan commit till Github. Buggar åtgärdas direkt av utvecklare. Saknas möjlighet att åtgärda direkt ska buggkommentar skrivas i samband med commit till github. Eventuella buggar ska alltid dokumenteras under respektive testfall och markeras antingen som åtgärdad eller återstående. Efter slutfört test med önskat resultat ska testfallet markeras och signeras som slutfört.

Omtestning – sker direkt i samband med buggfix.

Systemtester – Efter varje avslutat testfall följer systemtester. Ansvarig för systemtester är testare. Rapport av eventuella buggar vidarebefordras till utvecklare och projektledare.

Kompabilitetstester – Efter varje avslutat testfall följer kompabilitetstester. Ansvarig för systemtester är testare. Rapport av eventuella buggar vidarebefordras till utvecklare och projektledare. Kompabilitetstester ska utföras i Firefox, Safari och Chrome.

Testresultat
Enhetstester/Integrationstester – resultatet skrivs ut i utvecklarkonsollen.

Användartester – resultatet skrivs ut eller visas i klientens fönster.

Gymnastikligan Testfall: Användare inloggad

Testfall 1 – Användare inloggad

Beskrivning: Systemet ska kunna hantera inloggning och när användaren är validerad visa de funktioner som behörigheten tillåter. Användarfel vid inmatning av användarnamn och lösenord ska hanteras på ett för systemet säkert sätt och för användaren ett informativt och hjälpande sätt. Resultatet av testerna skrivs ut I konsollen.
Villkor - före: Användaren har ett användarkonto.
Villkor - efter: Användaren är inloggad I systemet och kan använda funktioner tillhörande behörigheten.
Testvillkor: Användarnamn och lösenord

Tester att genomföra:

Primärt flöde:
Användarnamn och lösenord är korrekta. Användaren kommer in I systemet och kan använda de funktioner som behörigheten tillåter.

Sekundärt flöde:
Användarnamnet och lösenord är fel.
Användarnamnet är fel.
Lösenordet är fel.

Gymnastikligan Testfall: Möjlighet att välja tävling och gren

Testfall 2 – Möjlighet att välja tävling att välja tävling och gren

Beskrivning: Användare med behörighet "domare" ska ha möjlighet att funktionen "poängsättning". När användaren valt funktionen "poängsättning" visas en lista med möjliga tävlingar att välja från. När tävling är vald visas en lista over möjliga grenar att poängsätta.
Villkor - före: Användaren är inloggad med rätt behörighet.
Villkor - efter: Användaren har valt tävling och gren att poängsätta och kommer automatiskt till listan över gymnaster.
Testvillkor: Behörighet "domare"

Tester att genomföra:

Primärt flöde:
Behörigheten är korrekt och användaren väljer funktionen poängsätta. En lista visas över möjliga tävlingar att välja från. Efter vald tävling kommer användaren automatiskt till listan "välj gren".

Sekundärt flöde:
Behörigheten är fel.
Användaren har rätt behörighet. Väljer tävling. Kommer vidare till "välj gren". Ångrar val av tävling och vill ändra.
Användaren har rätt behörighet. Väljer tävling. Väljer gren. Ångrar val av gren och vill ändra.

Gymnastikligan Testfall: Möjlighet att bekräfta deltagare

Testfall 3 – Möjlighet att bekräfta deltagare

Beskrivning: Användare med behörighet "domare" ska ha möjlighet bekräfta deltagarlistan vid en tävlingsgren.
Villkor - före: Användare har valt tävling och gren att poängsätta.
Villkor - efter: Användaren har bekräftat deltagarlistan och kan poängsätta enskilda gymnaster.
Testvillkor: Tävling är vald. Gren är vald. Deltagarlista finns.

Tester att genomföra:

Primärt flöde:
Användaren godkänner listan over deltagare.

Sekundärt flöde:
Användaren vill ta bort en deltagare/flera. Bekräftar därfter deltagarlistan.

Gymnastikligan Testfall: Möjlighet att bekräfta deltagare

Testfall 4 – Möjlighet att bekräfta deltagare

Beskrivning: Användare med behörighet "domare" ska ha möjlighet bekräfta deltagarlistan vid en tävlingsgren.
Villkor - före: Användare har valt tävling och gren att poängsätta.
Villkor - efter: Användaren har bekräftat deltagarlistan och kan poängsätta enskilda gymnaster.
Testvillkor: Tävling är vald. Gren är vald. Deltagarlista finns.

Tester att genomföra:

Primärt flöde:
Användaren godkänner listan över deltagare.

Sekundärt flöde:
Användaren vill ta bort en deltagare/flera. Bekräftar därfter deltagarlistan.

Gymnastikligan Testfall: Möjlighet att spara deltagarpoäng

Testfall 5 – Möjlighet att spara deltagarpoäng

.
Beskrivning: Användare med behörighet "domare" ska ha möjlighet sätta och spara poäng per deltagare.
Villkor - före: Användare har bekräftat deltagarlistan.
Villkor - efter: Användaren har sparat individuella resultat.
Testvillkor: Tal med två decimaler.

Tester att genomföra:

Primärt flöde:
Användaren skriver in poäng för enskild deltagare och klickar spara. Poängen sparas. Deltagaren markeras som sparad.

Sekundärt flöde:
Användaren skiver in poäng och sparar. Användren vill ändra poängen efter att ha sparat.
Användaren skriver in poäng som anses vara ogiltigt tal.

Testfall 6 – Möjlighet att avsluta gren

.

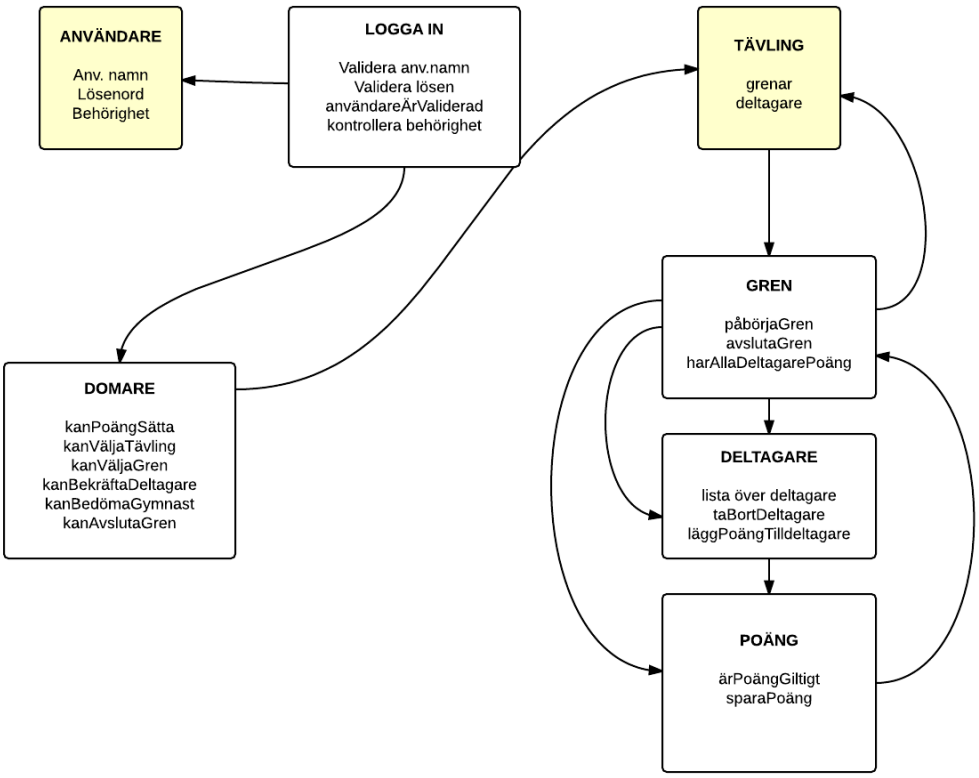
Beskrivning: Användare med behörighet "domare" ska ha möjlighet att avsluta tävlingsgren när alla gymnaster är klara.
Villkor - före: Användare har en korrekt deltagarlista och sparat poäng för alla deltagare.
Villkor - efter: Användaren har avslutat tävlingen och resultaten bokförs.
Testvillkor: Tal med två decimaler. Antal resultat överensstämmer med antal deltagare.

Tester att genomföra:

Primärt flöde:
Användaren klickar "avsluta tävling". Bekräftar att tävlingen ska avslutas. Resultaten bokförs.

Sekundärt flöde:
Användaren klickar "avsluta tävling". Bekräftar inte utan avbryter.
Användaren klickar avsluta tävling. Poäng saknas för någon eller några deltagare. Tillbaka till poängsidan.

Klassdesign



Testfall: testfall id 1 – Användare inloggad

Testfixturer: Lista med giltiga användare: [{user:"Johanna", pwd:"pwd1", authority:"Domare"},{user:"Anna", pwd:"pwd2", authority:"Gymnast"}],

Skrivit kod för att hantera ett användarnamn (mapp username)
Enhetstester:
1. Skickade in fel anv.namn. Konsollen skriver ”Inkorrekt användarnamn”.
2. Skickade in rätt anv.namn. Konsollen skriver ”Korrekt användarnamn”.
3. Skickade in tom sträng. Konsollen skriver ”Inkorrekt användarnamn”.

Skrivit kod för att hantera ett lösenord(mapp password)
Enhetstester:
4. Skickade in fel lösenord. Konsollen skriver ”Inkorrekt lösenord”.
5. Skickade in rätt lösenord. Konsollen skriver ”Korrekt lösenord”.
6. Skickade in tom sträng. Konsollen skriver ”Inkorrekt lösenord”.

Justering av enhetstester :
Kompletterade ovanstående med en kontroll av rätt anv.namn/lösenord, men skrivet med versaler istället. **Omtest.**

Slog ihop ovanstående enheter (mapp user)
Integrationstest:
1. Fel användarnamn, Fel lösenord. Konsollen skriver ”Inkorrekt anv.namn, korrekt lösenord”.
2. Tom sträng på båda. Konsollen skriver ”Inkorrekt anv.namn, inkorrekt lösenord”.
3. Rätt användarnamn, rätt lösenord. Konsollen skriver ”Korrekt anv.namn, inkorrekt lösenord”.
4. Rätt användarnamn, rätt lösenord men med versaler. Konsollen skriver ”Korrekt anv.namn, korrekt lösenord”.
5. Rätt användarnamn, fel lösenord. Konsollen skriver ”Korrekt anv.namn, inkorrekt lösenord”.
6. Rätt lösenord, fel användarnamn. Konsollen skriver ”Korrekt anv.namn, inkorrekt lösenord”.

Justering av integrationstest:
Kompletterade ovanstående med tom sträng på anv.namn, rätt lösenord. **Omtest.**

Skrivit kod för att hantera validering av användare (mapp isValidated)
Enhetstester
1. Fel användarnamn, Fel lösenord. Konsollen skriver ”Användaren finns inte”.
2. Tom sträng på båda. Konsollen skriver ”Användaren finns inte”.
3. Rätt användarnamn, rätt lösenord. Konsollen skriver ”Användaren är validerad”.
4. Rätt användarnamn, rätt lösenord men med versaler. Konsollen skriver ”Användaren finns inte”
5. Rätt användarnamn, fel lösenord. Konsollen skriver ”Användaren finns inte”
6. Rätt lösenord, fel användarnamn. Konsollen skriver ”Användaren finns inte”.
7. Rätt lösenord, tomt användarnamn. Konsollen skriver ”Användaren finns inte”.

Lade till validering av användarfunktionen (mapp user)
Integrationstest:
Testade alla enhetstester tillsammans och att konsollen skrev ut enhetstesternas meddelanden. Sammanslagningen skapade inga nya buggar.

Skrev kod för behörighet och utökade valideringsfunktionen med den koden (mapp user)
Enhetstest:
1. Testade att konsollen skrev ut behörighet i de fall användaren är validerad.

Skapade en egen klass för users och separerade loginfunktionen för att göra testerna mer dynamiska (mapp extendedUser).
Enhetstester:
1. Kontrollerade att listan med användare hade korrekt innehåll.
2. Kontrollerade att konsollen skev ut rätt behörighet när användaren är validerad.
3. Kontrollerade att konsollen skrev ut rätt meddelande när användaren inte är validerad.

Kompabiltetstest:
Testat att resultatet är detsamma i Firefox, Chrome och Safari. Fungerar likadant i alla tre.

Kommentarer till testsvit:
När jag började skriva kod, inte testerna utan den koden som skulle bygga upp loginfunktionen, föll det sig naturligt att köra enhetstester allt eftersom. När två bitar såsom användarnamn och lösenord var testade var för sig föll det sig lika naturligt att inte skapa fler små delar utan att testa de två tillsammans först. Jag gjorde sen enhetstester på behörigheten och la till den till de två andra. Byggde på loginfunktionen och körde integrationstest på det. Efter det blev det mer effektivt att bygga på den sammalagda koden och testa allt tillsammans efter hand istället för att testa enskilt.

Reflektioner

Jag kan definitivt se nyttan med att skriva testfallen före man sätter igång att skriva koden. Det blev mycket tydligare vilka alternativa flöden man behövde ta hänsyn till när programmeringen väl var påbörjad. Något som också slog mig var hur fokusen på vad som kan gå fel/bli fel ökade och det blev en styrande faktor.

Nu kände jag mig begränsad i min erfarenhet av programmering och skulle helst kanske satt upp en databas med användare och testat mot eller lagt till fler funktioner så som att användaren bara kan skriva fel lösenord tre gånger. I det här fallet blev det bara för rörigt för mig att jag istället försökte koncentrera mig på att bygga upp fallet och de tester kan klarade av.

Det har varit en utmaning att förstå vad ni är ute efter på varje labbdel och då har jag ändå använt min handledning för att få bättre grepp. Trots att handledaren var mycket hjälpsam och kunnig så är det här ett ämne som kan ta sig olika skepnader beroende på vem man talar med. Mer tydlighet i vad som efterfrågas vore uppskattat.

Jag hade också svårt att dela upp mitt kodarbete i de enskilda delar som labben efterfrågade. Jag tyckte de flöt ihop och gick in i varandra.

Planering/tidslogg labb 3

Uppgift	Tidsåtgång	Dag	Avvikelse
Förberedelser: Titta igen på två föreläsningar.	3 timmar	lördag 13/12	2 timmar. Hittade inte en FL, men tittade på den extrainspelade och spolade lite fram och tillbaka.
Förberedelser: Läsa 2 kapitel i boken.	3 timmar	söndag 14/12	7 timmar. Hade svårt att få grepp om det här. Googlade en massa. Gick över på måndagen.
Tidsplanering	0.5 timmar	måndag 14/12	0.5 timmar
Handledning	0.5 timmar	onsdag 17/12	0.45 timmar
Skriva om användningsfall efter feedback från handledare labb 2.	1 timme	måndag 15/12	1 timme
Testplan	1.5 timme	tisdag-onsdag 16-17/12	2 timmar. Fick läsa igenom lite olika fall för att hitta en nivå att lägga mig på. Tog mindre tid att skriva mer tid att få grepp om vad jag skulle ha med.
Testfall	2 timmar	tisdag-onsdag 16-17/12	1.30 timmar. Efter allt funderande hade jag det i stort sett klart i huvudet.
Klassdesign	1.5 timmar	onsdag 17/12	1 timme.
Kod	4 timmar	onsdag 17/12	se nedan
Enhetstester inkl omtester	2 timmar	torsdag 17/12	se nedan
Integrationstester inkl omtester	1 timme	tordag 17/12	Hade svårt att dela upp det här som beskrivet i laborationen. Tyckte att delarna föll in i varandra. Totalt kodande 6 timmar
Motiveringar och förklaringar	1timmar	torsdag 17/12	0.45 timmar
Reflektioner	1 timme	torsdag 17/12	0.5 timmar