Kvalitetsplan

Version 1.2 PUM02 2025-03-13

Projektidentitet

Medlemmar

Namn	Ansvarsområde	Mail
Alice Almgren	Teamledare	alial202@student.liu.se
Anton Taber	Testledare	antta671@student.liu.se
Axel Berg	Arkitekt	axebe390@student.liu.se
Isabel Neubauer	Dokumentansvarig	isane541@student.liu.se
Jakob Tormalm	Analysansvarig	jakto054@student.liu.se
Samuel Tuvstedt	Backendutvecklingsledare	samtu593@student.liu.se
Samuel Åkesson	Konfigurationsansvarig	samak519@student.liu.se
Simon Gunnarsson	Frontendutvecklingsledare	simgu061@student.liu.se
Stina Åström	Kvalitetssamordnare	stias606@student.liu.se

Kund:
Digitaliseringsavdelningen, Linköpings universitet

Handledare:
Eric Ekström

Examinator / kursansvarig:
Lena Buffoni

Dokumenthistorik

Version	Datum	Ändringar	Granskad av	Utfärdat av
1.2	2025-04-X	Lade till detaljer kring OpenAPI, Swagger och ReDoc	A.A, S.T	S.Å.
1.1	2025-03-13	Uppdaterats efter feedback från handledare	A.A, S.T	S.Å.
0.1	2025-02-21	Skapats	Alla	I.N., S.T., S.Å., S.G., S.Å.

Innehåll

1. Introduktion

Kvalitetsplanens syfte är att definiera de processer och aktiviteter som gör det möjligt för projektgruppen att samla in välgrundade bevis på att tidsredovisningssystemet uppfyller sina fastställda krav. Den säkerställer att processerna etableras, hanteras och underhålls under projektets gång av gruppmedlemmarna samt att aktiviteterna är anpassade efter produktens risknivå. Processerna utvärderas, utvecklas och förbättras kontinuerligt under projektets gång för att förbättra projektgruppens arbete. Kvalitetsplanen bidrar också till att säkerställa att produkten håller hög kvalitet i enlighet med standarden IEEE 730-2014 [1].

1.1. Definitioner, akronymer och förkortningar

Förklaringar till de förkortningar, definitioner och akronymer som används i det här dokumentet finns i projektets *begreppslista*[2].

2. Översikt av kvalitetsplan

Den här sektion innehåller en översikt av processer och introducerar organisation, beslutsfattande och dokumentation.

2.1. Organisation och ansvar

Kvalitetssamordnaren har huvudansvaret för kvalitetsarbetet i projektet. Det inkluderar att upprätta och underhålla kvalitetsplanen samt säkerställa att fastställda aktiviteter och processer genomförs korrekt. En central del av kvalitetsarbetet innefattar även att se till att processer och aktiviteter kontinuerligt utvärderas vilket innebär att förbättringsförslag och problem identifieras, diskuteras och sammanställs. Kvalitetssamordnaren ansvarar även för att genomföra och dokumentera de överenskomna förbättringarna, vilket säkerställer att processerna kontinuerligt utvecklas och förbättras. Dessutom ansvarar den för att identifiera bristande kunskap i kvalitetsarbete hos projektmedlemmarna, och tar beslut för att förbättra denna kunskap genom interna utbildningar.

Kvalitetssamordnaren delegerar kvalitetsarbetet till samtliga projektmedlemmar. Alla projektmedlemmar har ett ansvar att följa de kvalitetsrelaterade aktiviteter som fastställts i det här dokumentet.

2.2. Beslutsfattande

Processen för beslutsfattande beskrivs i projektgruppens *gruppkontrakt* [3], där det framgår hur beslut fattas och vilka frågor som bör diskuteras i helgrupp.

Bland annat anges att den ansvariga inom ett specifikt område har större beslutsfrihet vid mindre viktiga beslut som endast rör det egna ansvarsområdet. Detta innebär exempelvis att kvalitetssamordnaren har större frihet att besluta om kvalitetsrelaterat arbete, så länge det inte rör generella beslut som påverkar hela projektet.

2.3. Kommunikation och samarbete

Hur kommunikation och samarbete sker inom gruppen beskrivs i *gruppkontraktet* [3] och *projektplanen* [4].

2.4. Dokumentation och standarder

För att säkerställa produktens kvalitet produceras följande dokumentation:

- **Projektplan** [4] är utformad i enlighet med materialet i kursen TDDC88 [5] och behandlar organisation, planering och risker.
- **Kravspecifikation** [6] följer IEEE Std 830 [7] och definierar systemets funktionella och icke-funktionella krav samt eventuella begränsningar.
- **Arkitekturdokument** [8] beskriver systemets övergripande struktur, komponenter och hur de samverkar.
- **Testplan** [9] följer IEEE Std 829 [10] och IEEE Std 29119 [11] och beskriver teststrategi, testfall och testmiljö.
- **Testrapport** [12], följer IEEE Std 829 [10] och IEEE Std 29119 [11] och är en uppföljning på *testplanen* som utvärderar testresultatet samt bedömer produktens kvalitet.
- **Kvalitetsplan** (detta dokument) följer IEEE Std 730 [1] och beskriver kvalitetsarbetet.

Alla dokument skrivs på svenska och följer en enhetlig mall framtagen av dokumentansvarig.

2.5. Standarder, rutiner och konventioner

Utöver dokumentation krävs läsbar kod för att säkerställa att produkten fortsätter hålla hög kvalitet. I syfte av läsbarhet och underhållbarhet ska koden följa några väldefinerade konventioner inom programmering. Koden ska även vara konsekvent formaterat.

2.5.1. Kod-konventioner

Några mer specifika konventioner som ska strävas efter i projektet är:

- Variabler och funktioner som rör samma område ska ha självklara och konkreta namn. Kod ska dessutom sträva åt att introducera så *få* begrepp som möjligt.
- Nästlade kodblock ska undvikas. Utom vid exceptionella undantag ska inte kod som har nästlande djupare än tre nivåer godtas.
  - ”Guard clauses” [13] ska användas för att undvika nästlande.
- Kod ska ha bra beteendelokalitet (eng. Locality of Behaviour) [14].
- Koden ska vara skriven på ett modulärt sätt.
  - Varje modul ska ha ett tydligt ansvarsområde
  - I C# ska `interface` nyttjas där det är lämpligt för att exponera gränssnitt och dölja konkreta implementationsdetaljer.
  - Moduler ska ha så lite beroende på varandras implementation som möjligt.
- All kod (namn och kommentarer) ska vara skriven på engelska

2.5.2. Kodstil och formatering

Kod ska följa samma stil och formatering för att underlätta läsbarhet. Några krav ställs därför på koden:

- Kodblock ska vara indenterade enligt deras djup av nästlande.
- Namn ska följa de språk-idiomatiska stilerna. Exempelvis:
  - Funktioner i C# använda *Pascal Case* [15]
  - I TypeScript ska globala konstanter vara `UPPER_CASE` enligt `ESLint` [16].
- All kod som tillförs `main` ska vara formaterad konsekvent. Konfigurationsansvarig ansvar att skapa alla konfigurationsfiler för formateringsverktyg.
  - För filer skrivna i Typescript, CSS, och HTML ska Prettier [17] användas för att formatera filerna.
  - För C# ska CSharpier [18] användas för att formatera filerna.

3. Produktkvalitet

För att säkerställa att slutprodukten uppnår god kvalitet har ett antal kvalitetskrav fastställts som finns att läsa i *kravspecifikationen*[6]. Vidare har också mätprogram och tillvägagångssätt för utvärdering framställts för att uppnå kvalitetskraven.

3.1. Mätprogram

För att säkerställa att kvalitetskraven uppfylls kommer projektgruppen att genomföra tester kontinuerligt på produkten, processer och resurser enligt ett strukturerat schema. Testningen sker i slutet av varje sprint, där tre personer väljs ut att ansvara för att genomföra och dokumentera resultaten för denna sprint, enligt ett rullande upplägg. För att bestämma vilka personer som är ansvariga för testerna används en lista med alla projektmedlemmar där ansvaret roterar enligt listan.

En del av testerna kommer inte vara möjliga förrän tillräcklig funktionalitet har implementerats, men tester ska påbörjas i första möjliga skede.

Prestandatester utförs genom mätningar i en utvecklingsmiljö för att säkerställa att LCP [19] inte överskrider 2,5 sekunder i minst 95% av testfallen.

För att mäta prestanda kommer verktyget lighthouse-ci [20] användas. En pipeline på Azure kommer sättas upp som genererar en sammanställning över alla *Web Vitals*-värden [21], bland annat LCP [19]. Under mätning kommer en konsekvent bandbredd användas, exempelvis emulera 4G-nätverk, för att minska varians.

Vid *användbarhetstester* får en eller flera utomstående personer genomföra ett antal tydligt definierade uppgifter i systemet. Efteråt fyller de i ett SUS-formulär med tio påståenden om deras upplevelse av systemet och betygsätter hur väl påståendet stämmer på en skala ett till fem.

För *tillförlitlighet* kommer kodtäckning att granskas genom automatiserade tester där en viss procent av koden omfattas av testerna. Dessa automatiska tester sker genom *statement coverage*, alltså att varje rad i koden testas minst en gång. Testresultaten dokumenteras och diskuteras under mötet då sprint retrospective genomförs för att hitta eventuella förbättringsområden.

För att mäta *effektivitet* kommer samtliga projektmedlemmar rapportera in sin arbetade tid samt en beskrivning av vad de gjort och när. Denna information ger även en tydlig bild över projektgruppens resursanvändning och status. Den sammanlagda arbetade tiden jämförs sedan med den förväntade tidsåtgången på 400 timmar och visualiseras i ett burndown-diagram.

3.2. Utvärdering av produktens kvalitet

Produktens acceptans kommer kontinuerligt att utvärderas genom det fastställda mätprogrammet. Mätresultaten ger en tydlig översikt över produktens kvalitet. Om någon av mätningarna ger ett resultat som inte uppfyller kvalitetskraven, ska detta prioriteras och åtgärdas i nästa sprint.

När produkten uppnår samtliga kvalitetskrav och projektgruppen har följt de processer som definierats i denna kvalitetsplan anses den vara accepterad ur ett kvalitetsperspektiv.

4. Processkvalitet

Scrum används som arbetsmetod och kommer kontinuerligt utvärderas och förbättras under projektets gång.

4.1. Beskrivning av Scrum

Scrum är en agil arbetsmetod där små tvärfunktionella team samarbetar för att utveckla en produkt i korta cykler, kallade sprintar [22]. I slutet av varje sprint presenteras en potentiellt levererbar produkt.

4.1.1. Roller i Scrum

Scrum innefattar vanligtvis tre huvudsakliga roller [23]:

- **Scrum master**: Fungerar som en coach och säkerställer att Scrum följs samt ser till att teamet arbetar effektivt.
- **Produktägare**: Representerar kunden och ansvarar för produktbackloggen.
- **Utvecklingsteam**: Består av ett antal utvecklare som producerar mjukvaran.

Inledningsvis kommer teamledaren anta rollen som Scrum master med anledning av att dessa roller har liknande uppgifter: att coacha teamet och driva arbetet framåt.

Produktägarens ansvar har fördelats mellan tre projektmedlemmar: frontendutvecklingsledaren, backendutvecklingsledaren och den dokumentansvarige. Syftet med denna fördelning är att möjliggöra en så detaljerad sprintplanering som möjligt. Genom samarbete mellan dessa roller kan detaljerade uppgifter och användarhistorier skapas inför sprintplaneringsmötet vilket säkerställer att projektets viktigaste delar behandlas.

I slutet av varje sprint beslutas det om rollfördelningen inför nästa sprint, baserat på sprint utvärderingen och nästa sprints huvudsakliga fokus. En ytterligare anledning till omfördelning av roller är att samtliga projektmedlemmar ska få möjligheten att testa på olika roller och ansvarsområden.

4.1.2. Aktiviteter i Scrum

Scrum bygger på korta sprintar som vanligtvis varar mellan en och fyra veckor [23]. Som utgångspunkt kommer projektgruppen att arbeta i sprintar som varar en vecka. Varje sprint innehåller ett antal återkommande aktiviteter som strukturerar upp arbetet. Dessa beskrivs nedan:

- **Sprintplanering:** I början av varje sprint bestämmer produktägaren tillsammans med teamet vad som kommer implementeras under denna sprint. Gruppen inleder med att gemensamt gå igenom de sprintuppgifter som produktägarna föreslagit och som berör samtliga gruppmedlemmar. Därefter delas gruppen upp i backend- och frontend-arbetsgrupper för att fokusera på mer specifika uppgifter inom respektive område. Alla uppgifter prioriteras sedan på en skala från ett till fyra, där fyra motsvarar högst prioritet. Slutligen samlas hela gruppen igen för gemensam återkoppling, tekniska diskussioner och bedömning av sprintens rimlighet.
- **Dagligt Scrum möte:** Ett kort, dagligt möte på ungefär 15 minuter då samtliga teammedlemmar uppdaterar varandra om sitt arbete och kan be om hjälp vid behov. Dagliga Scrum-möten hålls online via Discord klockan 11 eftersom projektmedlemmarna

oftast har rast den tiden och har därmed möjlighet att delta. Varje projektmedlem ska svara på följande frågor:
  - Vad har du gjort sedan senaste mötet?
  - Vad ska du göra fram till nästa möte?
  - Har du fastnat på något och behöver hjälp?

- **Sprint review:** Ett möte som hålls i slutet av varje sprint. Produkten inspekteras och produktbackloggen ändras vid behov. Utvecklingsteamet demonstrerar arbetet, besvarar frågor, diskuterar vad som gick bra, problem de stötte på och hur dessa löstes. Teamet diskuterar även vad nästa steg i implementeringen är.
- **Sprint retrospective:** Ett möte som hålls sista dagen av sprinten, efter sprint review. Teammedlemmarna diskuterar resultatet, lärdomar och tar med sig detta inför planeringen av nästa sprint.

Med anledning av att sprintarna endast är en vecka beslutade projektgruppen att slå ihop sprint review och sprint retrospective till ett gemensamt möte för att effektivisera arbetet.

4.1.3. Verktyg i Scrum

Följande verktyg används i Scrum för att organisera arbetet [23]:

- **Produktbacklogg:** En ordnad lista som hanteras av produktägaren och innehåller de krav och användarhistorier som är kvar att implementera. Produktbackloggen baseras på de milstolpar och aktiviteter som beskrivs i *projektplanen* [4].
- **Sprintbacklogg:** En ordnad lista av de uppgifter som ska göras under den nuvarande sprinten. Uppgifterna ska vara relativt små så att utvecklarna vet vad som ska göras.
- **Burndown chart:** En visuell representation av det återstående arbetet i den nuvarande sprintens backlogg.

4.1.4. Kanban

För att strukturera upp arbetet ytterligare kommer metoden Kanban användas. Syftet med Kanban-tavlan är att visualisera arbetsflödet och på så sätt ge en tydlig överblick av arbetsfördelningen och framstegen. Verktyget Azure DevOps kommer att användas för att visualisera och underhålla brädet.

Den innehåller smarta funktioner, såsom färgkodning utifrån prioritet, möjlighet att tilldela ansvariga personer samt att lägga till kommentarer och annan relevant information. Tavlan är uppdelad i olika kolumner som motsvarar olika faser. Projektgruppen har beslutat att inledningsvis använda följande fyra kolumner:

- **Att göra:** Uppgifter som ingår i sprintbackloggen men som ännu inte påbörjats.
- **Pågående:** Uppgifter som är under arbete men ännu inte färdigställda.
- **Granskning:** Uppgifter som är färdigimplementerade men ska granskas av en annan projektmedlem och eventuellt testas.
- **Godkänd:** Uppgifter som är slutförda och godkända.

Till en början innehåller ”Att göra”- kolumnen uppgifterna från sprintbackloggen. Under sprintens gång kommer uppgifterna förflyttas genom faserna. Ifall det finns sprintuppgifter som inte slutförts, flyttas dessa över till nästa sprintbacklog.

4.2. Ansvar för initiering och utvärdering av processer

Kvalitetssamordnaren ansvarar för att de processer som projektgruppen beslutat om initieras och att samtliga processer utvärderas kontinuerligt. Samtliga projektmedlemmar har gemensamt ansvar att delta i utvärderingar samt vid initiering av nya processer. Beslut om att införa en ny process fattas i enlighet med gruppkontraktet [3]. Scrum master ansvarar för att nya sprinter initieras och utvärderas.

4.3. Utvärdering av processer

I slutet av varje sprint samlas projektgruppen och utvärderar samtliga processer under mötet för sprint retrospective. Både positiva och negativa aspekter ska diskuteras och dokumenteras. Följande frågor ska användas som utgångspunkt för diskussionen:

- Vad gick bra respektive dåligt under sprinten?
- Vilka problem stöttes på och hur löstes dessa?
- Vad kan förbättras till nästa sprint?

Insikterna från utvärderingen ska användas som underlag vid planeringen av nästa sprint. Kvalitetssamordnaren ansvarar för att genomföra de förbättringar som beslutas under mötet om inget annat bestäms.

4.3.1. Formulär

För att kontinuerligt utvärdera och förbättra projektgruppens arbetssätt inom Scrum kommer en systematisk utvärdering att genomföras efter varje sprint. Detta görs genom en standardiserad enkät där varje medlem anonymt får bedöma sprintens genomförande utifrån tydlighet i sprintmål, samarbete, leverans, arbetsbelastning och påverkan av föregående sprint retrospective som bedömningsgrund. Enkäten består av skalfrågor där en etta representerar missnöje och en femma nöjdhet. Insamlade data analyseras för att identifiera förbättringsområden. Resultaten diskuteras vid nästkommande sprint retrospective och används för att implementera åtgärder i kommande sprintar. Följande frågor används inledningsvis:

- Var sprintmålen tydliga och realistiska?
- Hade teamet en god kommunikation och samarbete under sprinten?
- Kunde de planerade uppgifterna levereras i tid och med god kvalitet?
- Sprint retrospective hjälpte oss att identifiera förbättringsområden och utvecklas som team.
- Arbetsbelastningen under sprinten var rimlig och hållbar.
- Jag är överlag nöjd med hur denna sprint genomfördes.

4.3.2. Andel slutförda uppgifter

I slutet av varje sprint, under mötet för sprint retrospective, sammanställs var varje uppgift befinner sig på Kanban-brädet. De procentuella andelarna för antal uppgifter i varje fas på Kanban-brädet beräknas för att ge en överblick av sprintens resultat. Resultatet diskuteras under mötet för sprint retrospective och används som underlag inför planeringen av uppgifter och arbetsmetoder i nästa sprint.

4.4. Förbättring av produkt och processer

I samband med utvärderingen av sprinten finns möjlighet att ta upp förbättringsförslag både gällande produkt och process.

Om en projektmedlem kommer på omfattande ändringar av produkten som kan förbättra resultatet, bör man:
1. Läsa *kravspecifikationen*, se om det fortfarande uppfyller alla krav.
2. Presentera idén till gruppen, förklara hur det förbättrar lösningen.

Därefter fattas ett beslut enligt de regler som fastställts i *gruppkontraktet*.
Kvalitetssamordnaren ansvarar för att genomföra de förbättringar som beslutas under mötet.

4.5. Automatisk dokumentation

I backend används .NET-paketet NSwag för att automatiskt dokumentera alla ändpunkter. På så vis får utvecklarna i frontend en lätt översyn över varje ändpunkts förväntad in- och utdata.

4.6. Processer för statisk analys av kod

All kod som ska tillföras till main-branchen görs så genom att en pull-request skapas. Minst två personer måste granska, lämna kommentarer och till sist godkänna koden innan den kan bli en del av main-branchen. Granskare väljs automatiskt och slumpmässigt för att säkerställa en objektiv kodgranskning.

På varje commit pushad till repo:t kommer statisk kodanalys automatiskt genomföras. Beroende på vilken del av kodbasen som ändrats kommer olika verktyg användas för analys.

- För frontend-kod skriven i TypeScript ska ESLint [16] användas.
- För backend-kod skriven i C# ska kompilatorns inkluderade statisk kodanalys-funktionalitet användas [24].

4.7. Dokumenthantering

För att säkerställa hög kvalitet på samtliga formella dokument som produceras under projektets gång ska alla stycken granskas av flertalet gruppmedlemmar. I det syfte ska varje korrekturläsare signera sina initialer när de har godkänt ett stycke.
Dokumentansvarig säkerställer att dokumenten ser bra ut innan inlämning. Om det finns feedback på innehållet eller språket i dokumentet, skriver projektmedlemmen sin feedback som en kommentar i texten, signerat med sina initialer. Ändringshantering i dokument sker sedan strukturerat genom dialog med den som skrev det berörda stycket.

4.8. Konfigurationshantering

För att varje medlem ska kunna arbeta på projektet ostört har konfigurationsansvarig ansvar att sätta upp en reproducerbar miljö som fungerar på Windows, Mac och Linux. Hela utvecklarmiljön ska också kunna köras i Docker-containrar [25].

4.9. Ändringshantering för kod

Eftersom main-branchen kommer vara skyddad, och ändringar endast tillförs genom pull-requests, kommer det inte ske konflikterande ändringar där. Konflikter kan dock uppstå om flera personer arbetar på samma feature-branch, men man kan tänka sig att de båda är överens och kan lösa mergekonflikter sinsemellan.

4.10. Bedöma projektmedlemmarnas färdigheter och kunskaper

Projektplanen beskriver projektmedlemmarnas färdigheter och kunskaper samt presenterar en plan för att förse medlemmarna med den utbildning som behövs för att genomföra projektet.

Samtliga projektmedlemmar har ett ansvar att fråga om hjälp om de känner att de saknar färdigheter och kunskaper. De som har kunskap ska då också stå till hjälp när andra medlemmar behöver det. Det är däremot viktigt att medlemmar tar ansvar och utbildar sig själva, alternativt ber om att få lämna över ansvar, om de gång-på-gång måste få hjälp.

För att kvalitetsarbetet genomförs på bästa möjliga sätt ska projektmedlemmarna utbildas i Scrum och DevOps.

5. Ytterligare överväganden

Nedan beskrivs ytterligare faktorer som kan påverka kvalitetsarbetet.

5.1. Undantag och avvikelser

Den enda godkända undantaget för att inte utföra en kvalitetsmätning är om kvalitetskravet ej är mätbart än eller om mätningen skulle ge ett missvisande resultat. Detta kan till exempel inträffa om funktionaliteten inte har implementerats fullständigt än, vilket skulle leda till att mätningen inte speglar den faktiska kvaliteten.

5.2. Riskhantering

Risker står definierade i *projektplanen* [4], som granskas och utvärderas varje vecka. Detta inkluderar att se över riskernas sannolikhet, påverkan och riskfaktor samt eventuellt ta bort eller lägga till nya risker.

5.2.1. Risker vid utförandet av kvalitetsarbete

En risk med kvalitetsarbetet är att projektmedlemmar kan uppleva att kvalitetsmätningarna tar onödig tid som kunde spenderats på ren kodning. För att undvika detta ska så många mätningar som möjligt automatiseras och övriga mätningar ska göras så effektiva som möjligt.

Processarbetet kan också ta mycket tid, vilket innebär en risk eftersom projektet har en begränsning på tid. För att undvika detta, utvärderas arbetet efter varje sprint. Varje projektmedlem har också ansvar att utbilda sig inom Scrum-arbetssättet, för att effektivare arbeta tillsammans med resterande i gruppen.

5.3. Testning

En *testplan* [9] har fastställts för att säkerställa att koden fungerar korrekt och är pålitlig.

6. Referenser

[1] ”IEEE Standard for Software Quality Assurance Processes”, *IEEE Std 730-2014 (Revision of IEEE Std 730-2002)*, nr , s. 1–138, 2014, doi: 10.1109/IEEESTD.2014.6835311

[2] A. Almgren *m.fl.*, ”Begreppslista”, technical report, jan. 2025.

[3] A. Almgren *m.fl.*, ”Gruppkontrakt”, technical report, feb. 2025.

[4] A. Almgren *m.fl.*, ”Projektplan”, technical report, jan. 2025.

[5] K. Sandahl, ”Project Management”, 24 september 2024. Tillgänglig vid: https://www.ida.liu.se/~TDDC88/theory/10project-management.pdf. [Åtkomstdatum: 16 februari 2025]

[6] A. Almgren *m.fl.*, ”Kravspecifikation”, technical report, feb. 2025.

[7] ”IEEE Recommended Practice for Software Requirements Specifications”, *IEEE Std 830-1998*, nr , s. 1–40, 1998, doi: 10.1109/IEEESTD.1998.88286

[8] A. Almgren *m.fl.*, ”Arkitekturdokument”, technical report, feb. 2025.

[9] A. Almgren *m.fl.*, ”Testplan”, technical report, feb. 2025.

[10] ”IEEE Standard for Software and System Test Documentation”, *IEEE Std 829-2008*, nr , s. 1–150, 2008, doi: 10.1109/IEEESTD.2008.4578383

[11] ”ISO/IEC/IEEE International Standard - Software and systems engineering — Software testing —Part 3: Test documentation”, *ISO/IEC/IEEE 29119-3:2013(E)*, nr , s. 1–138, 2013, doi: 10.1109/IEEESTD.2013.6588540

[12] A. Almgren *m.fl.*, ”Testrapport”, technical report, feb. 2025.

[13] Refactoring Guru, ”Replace Nested Conditional with Guard Clauses”, 11 december 2024, *Refactoring Guru*. Tillgänglig vid: https://refactoring.guru/replace-nested-conditional-with-guard-clauses. [Åtkomstdatum: 12 februari 2025]

[14] C. Gross, htmx.org, ”Locality of Behaviour (LoB)”, 29 maj 2020, *htmx.org*. Tillgänglig vid: https://htmx.org/essays/locality-of-behaviour/. [Åtkomstdatum: 12 februari 2025]

[15] Microsoft, ”C# language documentation”, *Microsoft*. Tillgänglig vid: https://learn.microsoft.com/en-us/dotnet/csharp/. [Åtkomstdatum: 10 februari 2025]

[16] ESLint, ”ESLint naming convention”, *ESLint*. Tillgänglig vid: https://typescript-eslint.io/rules/naming-convention/. [Åtkomstdatum: 14 februari 2025]

[17] Prettier, ”What is Prettier?”, *Prettier*. Tillgänglig vid: https://prettier.io/docs/. [Åtkomstdatum: 14 februari 2025]

[18] CSharpier, ”CSharpier”, 11 februari 2025, *CSharpier*. Tillgänglig vid: https://csharpier.com/. [Åtkomstdatum: 14 februari 2025]

[19] Google, ”Largest Contentful Paint (LCP)”, *Google*. Tillgänglig vid: https://web.dev/articles/lcp. [Åtkomstdatum: 21 januari 2025]

[20] Google, ”lighthouse-ci”, *Google*. Tillgänglig vid: https://github.com/GoogleChrome/lighthouse-ci. [Åtkomstdatum: 17 februari 2025]

[21] Google, ”Web Vitals”, *Google*. Tillgänglig vid: https://web.dev/articles/vitals. [Åtkomstdatum: 20 februari 2025]

[22] K. Sandahl, ”Processes and Lifecycles”, 26 februari 2025. Tillgänglig vid: https://www.ida.liu.se/~TDDC88/theory/11SLC-shown.pdf. [Åtkomstdatum: 24 september 2024]

[23] B. B. Frank Tsui Orlando Karam, *Essentials of Software Engineering*, Third edition. Jones, Bartlett Learning, 2014, s. 345.

[24] Microsoft, ”Overview of .NET source code analysis”, *Microsoft*. Tillgänglig vid: https://learn.microsoft.com/en-us/dotnet/fundamentals/code-analysis/overview. [Åtkomstdatum: 17 februari 2025]

[25] Docker, ”What is Docker?”, *Docker*. Tillgänglig vid: https://docs.docker.com/get-started/docker-overview/. [Åtkomstdatum: 17 februari 2025]