Linköpings universitet | Institutionen för datavetenskap
Examensarbete på grundnivå, 15hp | Datateknik
2024 | LIU-IDA/LITH-EX-G--24/015--SE

Bitech. Ett front-end-projekt

– En digital lösning för byggarbetsplatsen.

Bitech. A front-end project
– A digital solution for the construction site.

Hemming Gong
Emelie Gustavsson
Sabrina Holmegren
Ramon Ibrahim
Shayan Kharaghani
Olle Mineur
Mervan Palmér
Roberto Wilnerzon

Handledare : Jonas Wallgren
Examinator : Kristian Sandahl

LINKÖPINGS UNIVERSITET

Linköpings universitet
SE–581 83 Linköping
013-28 10 00 , www.liu.se

Upphovsrätt

Copyright

Sammanfattning

Den här rapporten beskriver resultaten och metodologierna för Bitech front-end-projektet, ett omfattande mjukvaruutvecklingsinitiativ som syftar till att digitalisera och effektivisera verksamheten på byggarbetsplatser. Projektet använder avancerade webbteknologier för att skapa användarvänliga, dynamiska och responsiva gränssnitt som är lämpliga för både platsbaserade och administrativa behov inom byggbranschen. Viktiga aspekter av projektet inkluderar integration av hantering av realtidsdata, mobilkompatibilitet för att säkerställa åtkomst på olika enheter, och robusta säkerhetsåtgärder för att skydda känslig projektdata. Under hela utvecklingsprocessen antog teamet agila metoder, med betoning på iterativ utveckling, kontinuerlig testning och regelbunden kundfeedback, vilket underlättade en flexibel och responsiv utvecklingsmiljö. Denna rapport ger en detaljerad analys av projektets utfall, lärdomar och rekommendationer för framtida initiativ som syftar till att öka den teknologiska adoptionen inom byggindustrin.

Nyckelord: Front-end, React, NextJS, Byggbransch, Agila metoder,
Webbutveckling, Webbapplikation, Modul-Vy-Komponent, Användarvänlig UX.

Tillkännagivanden

Vi vill rikta ett stort tack till vår kund, Bitech, för deras samarbete och tro på oss genom projektet. Ett särskilt tack till Björn Jönsson, som alltid funnits tillgänglig när vi haft frågor och funderingar. Vi vill även tacka vår handledare Jonas Wallgren som under projektets gång har bidragit med stöd och vägledning.

Innehåll

Figurer

Tabeller

Definitioner

- **Applikation** - Syftar i de flesta fallen på systemet som projektgruppen har utvecklat.
- **Git** - System för versionshantering av kod.
- **Google Sheets** - En webbapplikation för att skapa och redigera kalkylblad online.
- **GUI** - *Graphical User Interface*. Ett samlingsnamn för de olika användargränssnitt som går att se på en skärm.
- **ID06** - Ett system för identifikation av arbetare. I detta projekt antas det att varje arbetare har ett unikt ID06.
- **JavaScript** - Ett programspråk som används främst för webbutveckling.
- **Overleaf** - Programvara för att skriva LaTeX.
- **Ramverk** - En plattform som programmerare kan utgå ifrån vid programmering. Ramverk kan exempelvis innehålla stödprogram eller kodbibliotek för att underlätta utveckling.
- **TypeScript** - Ett programspråk baserat på JavaScript men med datatyper.
- **<html>** - Roten av ett HTML-dokument, som omger allt innehåll och anger att dokumentet är skrivet i HTML.
- **<head>** - Innehåller metadata om dokumentet, såsom titel, länkar till stilar, och skript.
- **<body>** - Innehåller allt innehåll som visas för användaren, såsom text, bilder, och andra element.
- **End-to-end-testning** - En metod som testar en applikations funktionalitet från början till slut under verkliga omständigheter.

1 Inledning

Denna rapport dokumenterar projektarbetet *Bitech front-end projekt*, som utförts av en grupp studenter vid Linköpings universitet inom ramen för kursen *Kandidatprojekt i programvaruutveckling*. Målet med projektet var att utveckla front-end för en webbapplikation som ska användas i byggbranschen. I detta avsnitt introduceras projektet och de frågeställningar som besvaras i rapporten.

1.1 Motivering

Samhället är i en tid där teknologi förekommer i allt större utsträckning och både formar och förändrar våra arbetsmetoder och tillvägagångssätt. Detta gäller inte minst inom byggindustrin där det ofta förekommer hantering av många ritningar och där ritningar kan tappas bort eller hamna hos fel aktörer. Kommunikationen mellan olika aktörer från byggarbetare till chefer och arkitekter kan vara bristfällig och det kan drabba arbetsflödet negativt. Missförstånd och förseningar kan uppstå, som i sin tur orsakar stora kostnader. Byggbranschen stod 2023 för 11 procent av Sveriges BNP [1] vilket gör att det finns stor potential för ekonomisk vinst. Dessutom finns många positiva effekter av att öka effektiviteten av byggprocessen. Detta är något som företaget Bitech arbetar med. I detta kandidatarbete presenteras en innovativ lösning för att optimera kommunikation och hantering av ritningar inom byggsektorn. Genom att erbjuda en välgrundad lösning till företaget Bitech strävar inte detta arbete bara efter att lösa problem utan det syftar även till att öppna dörrar för framtida implementationer.

1.2 Syfte

Projektets mål är att utveckla en lösning som förbättrar effektiviteten och därmed bidrar till kostnadsminskningar inom byggsektorn. Kommunikationen mellan olika parter som är involverade i en byggverksamhet ska underlättas och ritningshanteringen ska både förenklas och göras säkrare.

Syftet med rapporten är att samla och utvärdera de erfarenheter som gruppen fått av genomförandet av projektet. Dessutom syftar rapporten till att besvara de frågeställningar som har identifierats.

1.3 Frågeställningar

I detta kapitel presenteras de frågeställningar som arbetats med och besvarats under projektet. Genom att tydligt definiera frågeställningar har navigeringen underlättats mot att identifiera lösningar och strategier för att möta kundens aktuella behov. Frågeställningarna är:

1. Hur kan hemsidan implementeras så att värde skapas för kunden?
2. Vilka erfarenheter kan dokumenteras från programvaruprojektet som kan vara intressanta för framtida projekt?
3. Vilket stöd kan man få genom att skapa och följa upp en systemanatomi?
4. Vilken inverkan kan ett digitalt ritningshanteringsprogram ha på byggindustrin?

1.4 Avgränsningar

Projektet hade en begränsning på 400 timmar per gruppmedlem där gruppen bestod av 8 gruppmedlemmar. Detta innebar att en del användbara funktionaliteter som diskuterats med kunden behövde uteslutas, exempelvis GPS-spårning, för att hålla projektet inom tidsramen. Det som levererades vid projektets avslut var därmed inte en färdig produkt utan en första version som behöver utvecklas vidare. Projektets fokus låg även på front-end-utvecklingen för produkten. Därmed behövde flera funktionaliteter ha ett gränssnitt medan själva back-end-funktionaliteterna uteblev. Gränssnittet skulle vara utformat för att vara användarvänligt och lättanvänt, även för dem med begränsad teknisk kompetens. Slutligen var produkten avsedd att användas på surfplattor med en skärmstorlek på 14 tum. Gränssnittet blev därmed anpassat efter touch-funktionaliteter.

1.5 Kontext

Genomförandet av projektet skedde som en del av kursen *TDDD96, Kandidatprojekt i programvaruutveckling*. Kursen syftade till att ge både teoretisk och praktisk förståelse för hur utvecklingen av programvara går till [2]. Detta uppnåddes genom att studenter arbetade tillsammans i slumpmässigt sammansatta grupper för att leverera en lösning till en kund. Som stöd fanns en handledare med erfarenhet från tidigare projekt som gruppen kunde vända sig till vid frågor och funderingar. Med denna handledare skedde handledarmöten regelbundet. Dessa skedde antingen fysiskt vid Linköpings universitet eller virtuellt via Microsoft Teams.

Som ett kurskrav förväntades varje gruppmedlem lägga totalt 400 timmar på projektet under en period på cirka 5 månader. Inom ramen för dessa 400 timmarna var samtliga delar av processen inräknade. Delarna var: deltagande på seminarier, föreläsningar, utvecklandet av programvaran och skrivandet av denna rapport. Varje gruppmedlem behövde dessutom anta en roll med specifika ansvarsområden. Detaljer om detta finns i avsnitt 2.2. Kurskraven omfattade vilka dokument som skulle skrivas. Utöver denna rapport skrevs även följande dokument:

- **Arkitekturdokument** - Beskriver den övergripande strukturen och designen av systemet, inklusive moduler och hur de interagerar med varandra. Det ger en bild av hur systemet ska implementeras och säkerställer en enhetlig och skalbar arkitektur.
- **Kravspecifikation** - Definierar detaljerat de funktionella och icke-funktionella kraven för den programvara som ska utvecklas. Beskriver även användningsfall och fungerar som en referenspunkt för utvecklingen av systemet. Kraven är framtagna i samråd med kunden och har prioriterats för att betona vikten av kravet uppfylls inom tidsramen för projektet.
- **Kvalitetsplan** - Ger en överblick av hur kvaliteten i projekt upprätthålls genom de olika dokumenten och den programvara som producerats. Definierar även hur granskningar ska ske för att säkerställa att beskrivningar av programvaran i dokumenten överensstämmer med den faktiska designen och strukturen som slutligen godkänns.
- **Projektplan** - En övergripande plan som fastställer målen, omfattningen, tidslinjen och resurserna för projektet. Skrevs i början av projektet för att skapa en tydlig plan att följa. De risker som kunde identifieras inom projektet listades tillsammans med förslag på hur gruppen kunde arbeta för att minska sannolikheten att de skulle ske.
- **Systemanatomi** - Beskriver systemets struktur på en detaljerad nivå. Kartlägger olika funktioner i systemet och visar vilka kopplingar som finns mellan dessa.
- **Testplan** - Definierar strategin för testning av systemet. Detta inkluderar de nivåer av tester som ska genomföras, hur testomgivningen är utformad och hur processen för olika typer av tester ser ut. Även tester för att verifiera uppfyllnad av de olika kraven i kravspecifikationen specificeras.
- **Veckorapport** - Skickas in till gruppens handledare varje vecka. Ska innehålla information om vad som gjorts senaste veckan och vad som planeras att göra nästkommande vecka. Eventuella problem som dykt upp diskuteras och om tester har genomförts ska deras resultat även redovisas i rapporten.

2 Bakgrund

I detta avsnitt beskrivs kundens bakgrund, gruppmedlemmarnas tidigare arbetserfarenheter och gruppens resurser.

2.1 Kunden Bitech

Bygg Informations Teknologi i Sverige AB, förkortat Bitech, agerade som kund i detta projekt där Björn Jönsson, Peter Hagström och Fredrik Lindeberg var gruppens kontaktpersoner. Bitech är ett svenskt startupföretag vars ambition är att utveckla ett system för digital hantering av ritningar på byggarbetsplatser. Dessutom har företaget Bitech tidigare varit delaktiga i kursen TDDD96, Kandidatprojekt i programvaruutveckling, med samma projekt. Detta projekt skapades för att bygga vidare på de visioner som Bitech har.

2.1.1 Vision

Problematik och strul kan lätt uppstå när flera pappersritningar läggs ut på ett och samma bord vid byggarbetsplatsen. Det blir lätt förseningar som skapas på grund av missförstånd och dålig kommunikation. Därmed såg Bitech stor potential till förbättringar på byggarbetsplatser och hade som ambition att genomföra förbättringarna. Visionen är att skapa ett standardiserat program som möjliggör en sammanhängande kedja från och med att en ritning skapas hela vägen tills att dess byggprocess är avklarad.

2.1.2 Projektuppdrag

Bitech hade som mål att skapa ett system som kunde lagra och versionshantera ritningar och sedan visa upp dessa på byggarbetsplatser. Detta skulle uppnås genom att lagra ritningarna på en server som sedan kommunicerar med surfplattor som arbetarna har tillgång till ute på bygget. Uppdraget gick ut på att utveckla en mjukvara som möjliggjör läsandet och öppnandet av ritningar, där projektgruppen fick ansvar för att utveckla front-end-delen av systemet. Ambitionen var dessutom att använda detta systemet för att lösa andra vanliga problem inom kommunikation, materialhantering, information, sekretess och säkerhet.

Genom samtal med kunden fick projektgruppen kännedom att en av grundarna till Bitech, Björn Jönsson, hade mer än 20 års erfarenhet inom byggbranschen. Därmed blev Björn Jönsson primär informationskälla gällande byggarbetsplatsen och målgruppen.

2.2 Projektgruppen

Projektgruppen bestod av 8 medlemmar som alla hade varsin ansvarsroll för att strukturera arbetet. Dessa roller listas tillsammans med tillhörande gruppmedlem i tabell 2.1.

Tabell 2.1: Gruppens roller

Namn	**Roll**
Hemming Gong	Arkitekt
Emelie Gustavsson	Testledare
Sabrina Holmegren	Analysansvarig
Ramon Ibrahim	Utvecklingsledare
Shayan Kharaghani	Dokumentansvarig
Olle Mineur	Konfigurationsansvarig
Mervan Palmér	Teamledare
Roberto Wilnerzon	Kvalitetssamordnare

De olika rollernas ansvarsområden var:

- **Arkitekt** - Ansvarar för och dokumenterar arkitekturen och valet av tekniker i projektet. Identifierar de komponenter och gränssnitt som används.
- **Testledare** - Tar fram en plan för hur testning ska gå till och vad som ska testas. Dokumenterar testningsprocessen och beslutar om produktens status.
- **Analysansvarig** - Ansvarar för kontakt med kunden. Arbetar med att ta fram de behov och önskemål som finns för produkten från kundens sida och förmedlar sedan dessa till övriga i gruppen.
- **Utvecklingsledare** - Ansvarar för detaljerad design samt leder och fördelar utvecklingsarbetet. Fattar beslut om utvecklingsmiljö och organiserar utvecklarens tester.
- **Dokumentansvarig** - Ansvarar för att det finns mallar till samtliga dokument i projektet och att det finns en rutin för hur dessa uppdateras. Ser till att det finns en logotyp som används genomgående och att leverans av dokument sker innan givna deadlines.
- **Konfigurationsansvarig** - Bestämmer vilka arbetsprodukter som skall versionshanteras och vad som ska ingå i en utgåva. Väljer och underhåller verktyg för versions- och konfigurationshantering och ser till att gruppen använder verktygen på rätt sätt.
- **Teamledare** - Leder gruppens arbete och ser till att samtliga mål uppfylls och att processen följs på ett korrekt sätt. Representerar även gruppen gentemot utomstående parter.
- **Kvalitetssamordnare** - Kartlägger gruppens tidigare kunskaper och erfarenheter för att arbetet ska kunna fördelas på ett gynnsamt sätt. Tar fram en process för att kontrollera och säkerställa produktens kvalitet och dokumenterar denna.

2.2.1 Tidigare erfarenheter

Samtliga gruppmedlemmar har studerat inom datatekniskt inriktade program i åtminstone 2 år. Gruppmedlemmarna har då fått arbeta med programvaruutveckling på universitetsnivå och har under denna tid fått erfarenhet från ett flertal projektarbeten. Utöver det är även alla gruppmedlemmar bekanta med att använda sig av webbsidor, gränssnitt och ramverk vilket är relevant för uppdraget.

Erfarenheter från tidigare projekt som gruppmedlemmarna tog med sig till detta projekt var många. Bland dessa var det till exempel att ha ett gruppkontrakt, att ha en öppen kommunikation och transparens med varandra och, viktigast av allt, att ha ett gemensamt mål. Då detta var ett större projekt än vad alla har haft tidigare var gruppen medveten om att vissa erfarenheter från tidigare projekt inte var anpassade för detta och därmed var öppenheten för ändringar av arbetssätt viktigt att ha i åtanke.

2.2.2 Resurser

I detta kapitel presenteras resurser som har varit betydande under utvecklingen av projektet. Dessa resurser sträcker sig över ett brett spektrum, från akademisk litteratur och onlinekällor till praktiska verktyg och ramverk. Resurserna listas nedan med en kort förklaring till hur de användes.

- **IEEE** - IEEE har varit den primära källan vid utformningen av kravspecifikationen, kvalitetsplanen samt testplanen. Kravspecifikationen har skrivits enligt standarden IEEE 830 [3], kvalitetsplanen har skrivits enligt standarden IEEE 730[4] och testplanen har skrivits enligt standarden IEEE 829 [5].
- **Föreläsningar** - Under kursens gång fick gruppen tillgång till användbar information om kandidatarbetet och mycket av dokumentationen som har givits på föreläsningarna.
- **Seminarier** - Gruppen har fått möjligheter att delta på seminarier för att utvidga kunskaperna.
- **Loggbok** - För att dokumentera hur många timmar som lades ned, och på vad, skapades en loggbok i Google kalkylark för att registrera alla timmar.
- **Discord** - Discord har varit den huvudsakliga plattformen för intern kommunikation och informationsdelning mellan gruppmedlemmarna i projektet.
- **Discordserver** - En Discordserver är en digital plattform där medlemmar kan kommunicera genom text- och röstkanaler. Den fungerar som ett virtuellt rum för gemenskaper att diskutera olika ämnen, dela information och samarbeta.
- **GitHub** - GitHub används för versionshantering av kod, automatisk testning av kod och Kanban-tavla. Till den automatiska testningen är det begränsad körtid på 2000 minuter.

3 Teori

I detta avsnitt presenteras den teori som använts för att bygga upp systemet. Redogörelsen nedan presenterar valda ramverk, metoder och verktyg.

3.1 HyperText Markup Language (HTML)

HyperText Markup Language, förkortat HTML, är ett märkspråk som används för att strukturera och presentera innehåll på webben [6]. HTML använder *taggar* för att definiera olika element på en webbsida som till exempel rubriker, stycken, bilder och länkar. HTML-dokumentet har en grundläggande struktur med element som <html>, <head> och <body>, och HTML-koden tolkas av webbläsaren för att visa innehållet på ett korrekt sätt.

3.2 Cascading Style Sheets (CSS)

Cascading Style Sheets, förkortat CSS, är ett stilspråk som används för att beskriva presentationen av ett dokument skrivet i ett märkspråk som HTML [7]. CSS används för att kontrollera layout, färg, typsnitt och andra visuella aspekter av ett webbdokument. CSS-regler består av en väljare och en deklaration, där väljaren identifierar vilket HTML-element som ska formateras och deklarationen anger hur det ska formateras.

3.3 JavaScript

JavaScript är ett prototyp-baserat, dynamiskt och svagt typat programspråk som främst används på klientsidan i webbläsare för att skapa interaktiva webbsidor [8]. På grund av att det är ett dynamiskt och svagt typat språk, innebär det att variabler inte behöver deklareras med en specifik datatyp. Språket har stöd för funktioner som första klassens objekt, vilket innebär att funktioner kan behandlas som variabler. Detta möjliggör att funktioner kan passeras som argument och returneras från andra funktioner, vilket i sin tur gör det enklare att skapa modulär och återanvändbar kod. Detta bidrar till att utveckla mer avancerad och flexibel programvara. Dessutom är JavaScript ett prototyp-baserat språk, vilket innebär att objekt

ärver egenskaper och metoder från prototypobjekt. JavaScript används ofta i samband med HTML och CSS för att utveckla webbsidor.

3.4 VueJs

VueJs är ett JavaScript-ramverk för att bygga användargränssnitt [9]. VueJs bygger på standard-HTML, CSS och JavaScript och erbjuder en deklarativ, komponentbaserad programmeringsmodell som hjälper utvecklare att effektivt skapa användargränssnitt. VueJs är ett progressivt ramverk som kan införlivas gradvis i befintliga projekt. Dess kärna underlättar integration med andra bibliotek och projekt. VueJs är lätt att lära sig, särskilt för utvecklare med befintlig kunskap i HTML, CSS och JavaScript. Ramverket är kapabelt att driva avancerade Single-Page-Application när det används tillsammans med moderna verktyg och stödbibliotek.

3.5 React

React är ett JavaScript-bibliotek som används för att bygga användargränssnitt vid webb- och applikationsutveckling [10]. React baseras på att mindre komponenter utvecklas som sedan kan återanvändas genom hela applikationen. Genom att följa denna utvecklingmetod blir koden modulär, samt säkerställer att utseende och funktionalitet stämmer överens i olika delar av applikationen. React ger en snabbare uppdatering av webbsidor genom att enbart uppdatera det som faktiskt ändras, istället för att hela webbsidan genereras på nytt vid varje ny uppdatering [11].

3.6 Olika typer av applikationer

När en webbsida implementeras finns det flera alternativ för hur sidan kan laddas in. Två av dessa metoder beskrivs nedan.

3.6.1 Multi-Page-Application (MPA)

Multi-Page-Application använder en traditionell webbapplikationsmodell där varje ny sida som användaren begär innebär att en helt ny sida laddas från servern [12]. Detta tillvägagångssätt är fördelaktigt när storskaliga webbplatser byggs med omfattande innehåll som kräver frekventa uppdateringar som e-handelssidor eller online-tidningar. Varje sida kan länka till ytterligare sidor med nytt innehåll istället för att dynamiskt uppdatera en enda sida. Denna struktur gynnas av förbättrade utvidgningsmöjligheter eftersom varje sida kan programmeras individuellt för att möta specifika behov.

3.6.2 Single-Page-Application (SPA)

Single-Page-Applications är utformade för att erbjuda en mer dynamisk användarupplevelse genom att ladda all nödvändig kod i HTML, JavaScript och CSS med en enda laddning av webbsidan. Dessutom kan en SPA dynamiskt uppdatera innehåll vid behov utan att uppdatera hela sidan [12]. SPA-arkitekturen minskar serverbelastningen, förbättrar prestandan och ger en bättre användarupplevelse då användare slipper klicka runt sig via länkar. Dock kan SPA möta utmaningar då allt innehåll en användare kan tänkas se måste visas på en enda sida.

3.7 TailwindCSS

TailwindCSS, ibland förkortat till Tailwind, är ett ramverk för CSS som tillåter utvecklare att skriva CSS direkt i JavaScript-koden istället för att behöva separata filer [13]. TailwindCSS

har inbyggda CSS-klasser som hjälper utvecklare att ha en kodbas som är mer konsekvent i namngivningen. Det blir då lättare för nya utvecklare att sätta sig in i koden. Dock är det en inlärningsfas för att lära sig TailwindCSS-klasserna. TailwindCSS ger även utvecklare möjligheten att kunna definiera egna klasser för att anpassa designen av eller utseendet för komponenter.

3.8 React Testing Library

React testing library är ett verktyg för att testa React-komponenter [14]. Det är utformat för att underlätta testning av komponenterna från en användares perspektiv. Det är därmed inte själva implementationen som testas, då denna kan variera, utan funktionaliteten står i fokus.

3.9 Jest

Jest är ett testramverk utvecklat för att testa kod skriven i JavaScript [15]. Jest har inbyggd funktionalitet för att testa applikationer skrivna i React. Det ger möjlighet att testa hur React-komponenter renderas och hur de reagerar på olika användarinteraktioner. Det går även att simulera externa moduler eller tjänster som komponenten beror på. Detta tillåter isolering vid tester av en enskild komponent, vilket gör att den kan testas utan påverkan av externa implementationer.

3.10 Cypress

Cypress är ett testramverk som används för att genomföra automatiska tester på webbapplikationer och har funktionalitet för att testa applikationer skrivna i React [16]. Möjlighet finns att simulera inskrivning av text, knapptryck och andra användarinteraktioner, vilket tillåter end-to-end-testning.

3.11 Lighthouse

Lighthouse är ett verktyg för att automatiskt testa hemsidors kvalitet [17]. Verktyget erbjuder en överblick med 4 sammanfattade betyg på en skala på 0-100. Kategorierna är *prestanda*, *tillgänglighet* och *bästa metoder*.

3.12 Scrum

Scrum är en agil arbetsmetod, innebär flexibel och iterativ utveckling med fokus på kundcentrering och kontinuerlig förbättring, som ofta används vid produktutveckling. Inom Scrum finns tre olika roller att utgå från [18]. Produktägare ansvarar för att representera de olika intressenterna och att definiera produktens krav och mål utifrån deras intressen. Scrum master ansvarar för att Scrum-processen följs och för att lösa de hinder som kan påverka gruppens framsteg. Teamet ansvarar för att planera, utveckla och leverera en fungerande produkt.

Det finns även en produktbacklog som är en prioriterad lista över alla de uppgifter som ska genomföras under projektets gång [18]. Arbetet delas upp i kortare, förutbestämda tidsperioder som kallas sprintar. Under en sprint tar gruppen sig an arbetsuppgifter från backlogen som den tror är möjligt att genomföra under den givna tiden för en sprint. Gruppen arbetar för att den funktionalitet de implementerar under sprinten ska vara fungerande vid sprintens slut. Detta möjliggör demonstration för intressenter och gör därmed att gruppen kan få regelbunden feedback från kunden.

Scrum inkluderar även flera regelbundna möten [18]. Varje dag genomförs ett stand-up-möte där gruppen går igenom det arbete som gjordes under gårdagen, planerar det arbete som ska genomföras under dagen och diskuterar eventuella hinder som identifierats. Inför varje sprint genomförs ett planeringsmöte där det bestäms vilka arbetsuppgifter som ska inkluderas. När en sprint avslutas genomförs även ett retrospektiv. Detta är ett möte för analys av sprinten gällande vad som fungerat bra, vilka problem som dykt upp och vilka åtgärder som kan vidtas för att undvika dessa problem i framtiden.

3.13 Parprogrammering

Parprogrammering är en programutvecklingsmetodik där två programmerare jobbar tillsammans vid en gemensam dator [19]. Den ena kallas för en *förare* och är den som skriver koden, medan den andra kallas för *navigatör* och granskar koden. Dessa två personer växlar ofta mellan rollerna för att få variation. Fördelar med detta arbetssätt är att föraren kan fokusera på den aktuella uppgiften, medan navigatören kan fokusera på förbättringar och framtida uppgifter eller problem. I studien baserad på studenter presterar studenter som använder sig av parprogrammering 18 procentenheter bättre än de som inte använder sig av tekniken [19]. Resultatet var baserat på funktionalitet och läsbarhet på koden som producerades.

3.14 Kanban

Kanban är en agil arbetsmetod som avser att visualisera och optimera arbetsprocessen [20]. För detta används främst en Kanban-tavla, vilket är en visuell tavla uppdelad i olika fält för att representera olika steg av arbetsflödet. Exempel på fält är ”Att göra”, ”Pågående” och ”Färdig”. För varje arbetsuppgift skapas ett kort som sätts på tavlan och som förflyttas mellan de olika fälten allteftersom arbetet fortlöper. Antalet pågående arbetsuppgifter begränsas dessutom för att minimera behovet att byta mellan olika arbetsuppgifter. Detta ökar effektiviteten då bytet mellan arbetsuppgifter tar tid.

3.15 Sustainability Awareness Framework (SusAF)

SusAF står för Sustainability Awareness Framework [21]. Det är en metod för att analysera de effekter som kan uppstå som en följd av implementering av ett mjukvarusystem. De identifierade effekterna delas upp i tre nivåer:

- Omedelbara effekter - uppstår som en direkt följd av systemets utvecklande eller användande.
- Möjliggörande effekter - uppstår över tid då systemet används.
- Strukturella effekter - bestående effekter som påverkar samhället i stor skala.

Effekterna kategoriseras även utifrån fem olika dimensioner: miljömässigt, socialt, individuellt, ekonomiskt och teknologiskt.

3.16 Next.js

Ett ramverk som är baserat på och utökar funktionaliteten hos React [22]. Next.js bidrar med en bättre användarupplevelse genom optimering av applikationen, bland annat genom att React-komponenter renderas på servern innan de skickas till webbläsaren. Utvecklingsfasen underlättas även genom att servern automatiskt startar om vid kodändringar. Next.js utökar även med extra funktionalitet gällande routing och navigation på en hemsida, där routing innebär hur användare skickas mellan olika sidor på en webbapplikation.

3.17 Figma

Figma är ett designverktyg som används för att snabbt och enkelt skapa prototyper av applikationer [23]. Tjänsten är molnbaserad och tillåter flera användare att arbeta tillsammans på samma projekt i realtid.

3.18 GitHub

GitHub är ett verktyg för versionshantering och implementerar *git* [24]. GitHub har flera funktioner för programvaruutveckling. Några vanliga funktioner är:

- *Repository*: Användare kan skapa en digital mapp för att lagra och organisera sitt projektarbete. Ett *repository* innehåller all kod, filer och resurser för ett specifikt projekt.
- Grenar: Används för att kunna arbeta parallellt på olika funktioner eller ändringar av en kodbas. Koden kan testas för sig själv innan det sammanfogas med resterande kodbas.
- *Commits*: Användare kan göra ändringar i koden och skicka dem till GitHub genom ”*commits*”. Varje *commit* representerar en enskild ändring eller uppsättning av ändringar.
- *Push*: Användare skickar sin commit till GitHub.
- *Pull förfrågan*: När en användare vill införa sina ändringar från en gren till huvudprojektet, skapar de en pull förfrågan. Andra användare kan granska ändringarna, ge feedback och godkänna dem in i huvudprojektet.
- Ärenden: GitHub tillhandahåller verktyg för att hantera och spåra buggar, önskade funktioner och andra ärenden.
- Samarbete: GitHub möjliggör samarbete mellan utvecklare genom att erbjuda verktyg för att diskutera, granska och bidra till varandras kod.
- Automatiska tester: GitHub möjliggör körning av automatiska tester. GitHub kallar det för *GitHub Actions*.

4 Metod

I detta avsnitt presenteras den metod som systematiskt använts för att besvara frågeställningar som formulerats i kapitel 1.3. Den detaljerade redogörelsen för den valda metoden och dess tillämpning, som presenteras nedan, belyser hur arbetet har framskridit för att uppnå de fastställda målen. Observera att projektarbetet inte följde processerna nedan i kronologisk ordning.

4.1 Erfarenhetsprocessen

Under erfarenhetsprocessen dokumenterades olika aspekter av projektets genomförande. I detta avsnitt presenteras dessa och går igenom den inverkan de har haft på projektet.

4.1.1 Tidsrapportering

För att hantera tidsrapporteringen i projektet valdes *Google Sheets* som det verktyget. Detta beslut motiverades av tidigare erfarenheter med programmet för liknande ändamål. Tillvägagångssättet implementerades genom en strukturerad dokumentation av den dagliga arbetsinsatsen. Den var uppdelad i tidsintervaller av maximalt 2 timmar för att möjliggöra en detaljerad överblick av genomförda aktiviteter. Varje tidsintervall som dokumenterades inkluderade även specificerad information kring arbetet som blev utfört. Denna process var en daglig rutin där varje gruppmedlems ansvarade för att noggrant registrera sina arbetsinsatser. Slutligen sammanställdes och överlämnades tidsrapporterna veckovis till handledaren.

4.1.2 Stand-up-möten

För projektets arbetsstruktur implementerades dagliga stand-up-möten som ett strategiskt verktyg för effektiv kommunikation och arbetskoordination. Dessa möten hölls korta och var avsedda för de medlemmar i gruppen som hade planerat att arbeta under den aktuella dagen. Dessa dokumenterades i en kanal i en gemensam Discordserver som var designerad för dessa möten och följde ett format bestående av datum, utfört arbete, planerade arbetsuppgifter och eventuella problem.

Dessa möten var inte obligatoriska utan var endast avsedda för de som hade för avsikt att arbeta under dagen, som nämndes ovan. Denna flexibilitet möjliggjorde en anpassning till individuella arbetsmönster då det fanns en medvetenhet att gruppmedlemmarna inte alltid kan arbeta under samma tid.

4.1.3 Scrum och Kanban

I kapitel 3.12 presenteras en ingående beskrivning av arbetsmetoden Scrum och dess funktion. Gruppen har under projektets gång baserat sitt tillvägagångssätt på dessa riktlinjer, samtidigt som egna anpassningar har implementerats för att motsvara projektets behov.

Gruppen utgick från en rad krav där gruppen valde att använda ett Kanban-inspirerat system som fungerade som produktbacklog istället för att använda strikta sprints som i den traditionella Scrummetoden. Uppgifter hanterades kontinuerligt och gruppen hade möjlighet att lyfta upp nya uppgifter när det behövdes.

4.2 Utvecklingsprocessen

För att säkra att projektet skulle nå sina mål införde gruppen en utvecklingsprocess. Den omfattar avgörande steg som kommunikation, versionshantering och testning.

4.2.1 Kommunikation

Nedan följer en beskrivning av den genomförda kommunikationen, både externt och internt, under projektets gång.

4.2.1.1 Extern kommunikation

Under projektets genomförande upprätthöll gruppen regelbunden kommunikation med kunden. Kommunikationen med kunden ägde huvudsakligen rum i antingen chatten Whatsapp, på plats i universitetets lokaler eller virtuellt genom Microsoft Teams.

4.2.1.2 Intern kommunikation

För den interna kommunikationen använde gruppen två olika kommunikationskanaler baserat på om frågan eller ärendet ansågs viktigt. Messenger användes som kommunikationskanal för att hantera snabba frågor och informella ärenden relaterade till projektet.

För att behandla mer formella frågor och möjliggöra distansmöten inrättade gruppen en Discordserver som en centraliserad kommunikationsplattform. Discordservern strukturerades noggrant med olika kanaler som var avsedda för specifika ärenden. Varje kanal på Discordservern tilldelades särskild information och diskussion relaterad till ett specifikt ämne eller en uppgift i syfte att skapa en lätthanterlig arbetsmiljö.

4.2.2 Prototypning

Utformningen av användargränssnittet skedde med hjälp av prototypningsverktyget Figma. Kvalitetssamordnaren och den dokumentansvarige utsågs med ansvaret för att skapa en första prototyp. Kunden fick möjligheten att lämna sina åsikter och eventuella ändringsförslag.

4.2.3 Versionshantering

Versionshanteringen av projektets kodbas sköttes genom ett *repository* på GitHub. Inom kodbasens utveckling och hantering implementerades tekniken *feature branches*, översatt till gre-

nar i 3.18, i gruppens repository. För att effektivisera dokumenthanteringen användes Overleaf, en webbplattform för samarbete och gemensam redigering av dokument.

4.2.4 Testning

Under projektets inledande fas skapades en testplan som inkluderade tester utformade för att säkerställa att samtliga krav i kravspecifikationen var uppfyllda. För projektet planerades följande fyra olika typer av tester.

Enhetstester planerades för att testa de minsta enheterna av koden, till exempel funktioner, klasser och komponenter för att säkerställa korrekt avsedd funktionalitet. *Integrationstester* planerades för att testa integrationen mellan olika enheter och därmed upptäcka eventuella problem i hur de interagerar med varandra. *Systemtester* planerades för att testa att hela applikationen fungerade korrekt i en realistisk miljö. Till sist planerades en *beställargranskning* för att kunden personligen skulle få möjlighet att testa applikationen. På så sätt kan det säkerställas att kundens förväntningar och krav är uppfyllda samt att applikationen är tillräckligt användarvänlig för målgruppen.

Följande fyra verktyg planerades att användas för att skriva dessa tester. *React Testing Library* användes för enhetstester [14]. *Jest* användes för kompletterande enhetstester och integrationstester [15]. *Cypress* användes för systemtester [16]. *Lighthouse* användes för att mäta prestanda, tillgänglighet och om utvecklad kod följde kodstandarder [17].

Som teststrategi användes botten-upp, där enhetstester genomfördes först, följt av integrationstester och till sist systemtester. För att automatisera tester användes *GitHub Actions* och de automatiska testerna skulle köra testverktygen Lighthouse och Cypress.

5 Resultat

I detta avsnitt presenteras de resultat som gruppen uppnåde vid projektets slut. Här behandlas resultat av gruppens erfarenheter, processer och systemimplementationen. Dessa resultat diskuteras sedan vidare i diskussionsavsnittet 6.

5.1 Systembeskrivning

I detta kapitel beskrivs systemets struktur och avgränsningar för ritningshanteringsprogrammet som implementerades i form av en webbapplikation. Programmet ansvarar för att visa upp ritningar i ett digitalt format i exempelvis png eller pdf. Detta program erbjuder en navigering som sköts av knappar utan behovet av gömda tangentkommandon. Man befinner sig som mest 8 knappklick ifrån sin destination och programvarans komponenter är anpassade efter att användaren arbetar på en surfplatta. Mer om detta i avsnitt 5.1.1. Systemet ser även till att flera ritningar kan öppnas samtidigt och sparar dessa i en lista där användaren kan bläddra mellan dem. Utöver det kan även ritningarna öppnas i fullskärmsläge.

Det finns även en *"display"* för hemsidan som all funktionalitet nås från ett historikflöde som är anpassat efter region och har funktionalitet för att visa dagens väder. Dessutom finns ett "nyhetsflöde" som visar aktuell information, incidenter och annat som är av intresse för användaren. Till slut finns även "dagens schema", som sorterar ärenden och händelser i ett tabellformat.

Det saknas däremot funktionalitet för back-end-delar av systemet som behövs för att bland annat möjliggöra en versionshantering och smidig överföring av ritningar. Därmed är adminsidan svagt utvecklad och dessutom finns inget GUI för att lägga nya ritningar i systemet. Fortsättningsvis designades ett taggsystem som skulle hantera indelningen av varje arbetares konto och gjorde det möjlig för systemet att kontrollera behörigheter, men detta hann inte implementeras i praktiken. Det finns dock en primitiv inloggningssida som i framtiden kan utvidgas så att byggarbetaren skannar sitt ID06-kort som kan användas för att logga in och automatiskt kopplas till rätt behörigheter. Till sist finns varken en sida för att skicka in formulär eller chattfunktionaliteten då de ej hunnits implementeras.

5.1.1 Användargränssnitt

Användargränssnittet är utformat för att vara användarvänligt med stora, tydligt märkta knappar med textförklaringar. Färgschemat använder hög kontrast mellan elementen för att säkerställa optimal läsbarhet i olika ljusförhållanden, speciellt vid direkt solljus utomhus. Detta gör att gränssnittet är användbart i mobila situationer där ljusförhållanden ändras. Navigeringspanelen har ett litet träddjup vilket innebär att användarna kan navigera genom funktionerna med ett litet antal steg. Detta ska minska den kognitiva belastningen och gör det lättare att snabbt hitta önskad funktion eller information, se figur 5.2-5.3. Dessutom är gränssnittet minimalistiskt för att motverka distraktioner och oklarheter.

5.1.2 Arkitektur

I utvecklingsprocessen för applikationen började gruppen ursprungligen med att använda en befintlig kodbas som hade utvecklats av tidigare kandidatgrupper. Initialt skapades en generell design för applikationen med hjälp av designverktyget Figma samt mer traditionella metoder som att rita på tavla och papper. Denna första fas fokuserade på att skapa en visuell och funktionell struktur baserad på den befintliga kodbasen.

Efter diskussioner med kunden och utifrån deras önskemål gjordes ett betydande strategiskt beslut att starta om utvecklingsprocessen med ett nytt tekniskt ramverk vilket innebar att all befintlig kod skulle skrivas om. Valet föll på React. Detta gav gruppen möjligheten att förbättra applikationens struktur.

När kandidatgruppen granskade den tidigare kodbasen och jämförde den med de nya kraven och möjligheterna som React erbjöd uppstod en idé om att inte bara uppdatera tekniken utan också att förbättra hela applikationens arkitektur. Trots att den generella designen, definierad i Figma och på papper, behölls i stora drag genomgick kodstrukturen en omfattande förändring för att utnyttja Reacts komponentbaserade arkitektur.

Denna nya arkitektur innebär att applikationen är uppbyggd av återanvändbara komponenter som kan hantera sitt eget tillstånd och logik, vilket effektiviserar både utvecklingsprocessen och underhållet av applikationen. Varje komponent är designad för att vara självständig och modulär vilket möjliggör enklare tester och bättre separation av funktionalitet av ansvar inom applikationen. Genom att använda moderna tillvägagångssätt som modulbaserad kod och tillståndshantering via *state management*, har gruppen kunnat skapa en mer robust, skalbar och underhållsvänlig applikation. Detta motsvarar väl kundens behov och förväntningar.

5.1.3 Design och systemanatomi

I detta kapitel beskrivs beslut som fattades kring designen och systemanatomin vilket resulterade i applikationens utformning. Genom att skapa och följa upp en systemanatomi kunde gruppmedlemmarna lättare få en översiktlig struktur för systemet vilket hjälpte gruppen att designa systemet med varandras arbete i åtanke. När systemet väl nått flera viktiga milstolpar i sitt utvecklingsprojekt användes även anatomin för att se möjligheter för vidareutveckling.

Centrala funktioner som ”Visa ritningar”, ”Nyhetsflöde”, ”Admin”, ”Display” är implementerade. Dessa implementerade delar markeras med grön enligt tabell 5.1 och utgör kärnan i systemet och skapar grunden för användarinteraktionen. Genom komponenter kan användare navigera i systemet och administratörer kan övervaka och hantera systemets drift. Förhoppningen för framtiden är att all datahantering ska vara centraliserad och säker enligt figur 5.1.

Andra funktioner vars implementation har påbörjats markeras med grå färg enligt tabell 5.1. Dessa är viktiga för applikationen och därmed har koden för exempelvis ”Hantera ritningar”, ”Språk”, och ”Materiallista” levererats till kunden för vidarutveckling. Föregående komponenter är avgörande för applikationens förmåga att presentera ritningar och materialinformation på ett effektivt sätt. Detta är användbart för användare som arbetar inom områden som tillverkning, konstruktion eller teknik. Detta då de ger en mer användarvänlig upplevelse med möjligheten att anpassa innehållet efter olika användarbehov och preferenser.

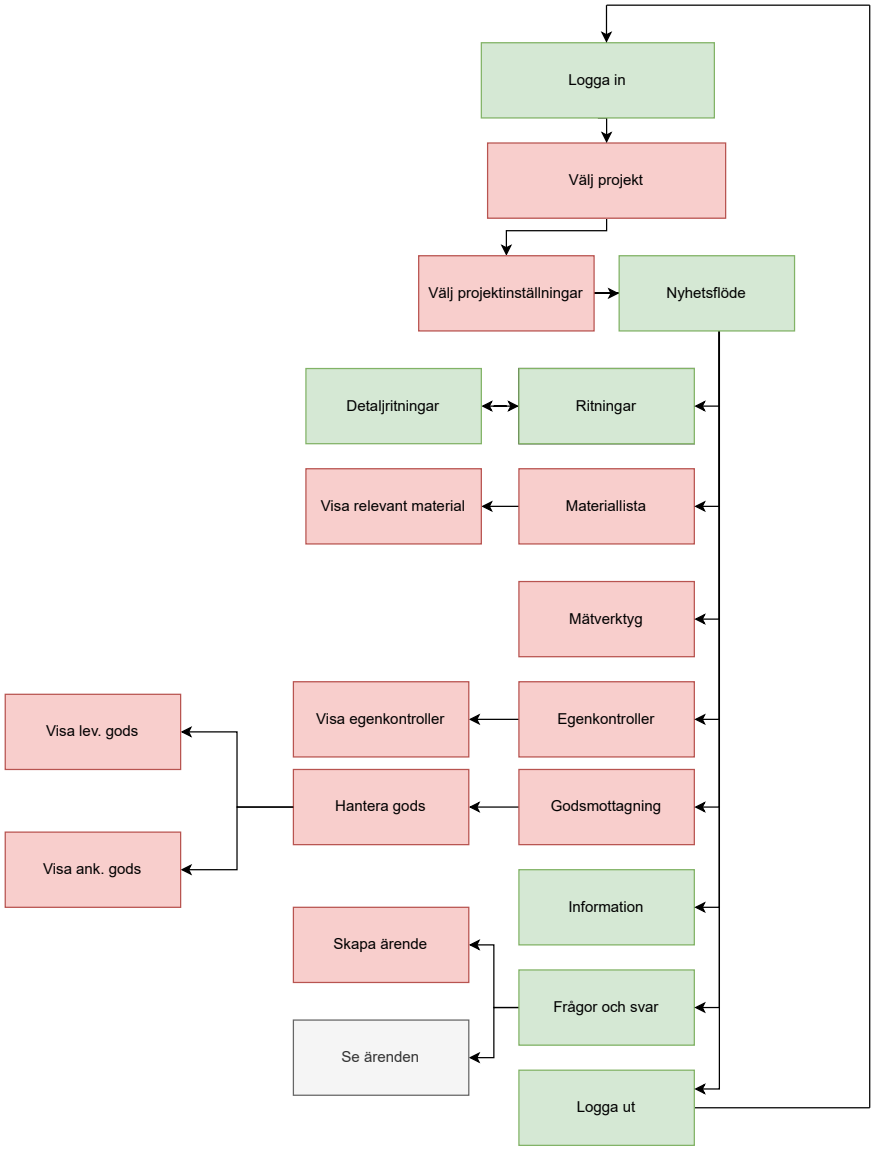
Vissa funktioner är fortfarande under utveckling som ”Chattfunktion”, ”Felanmälan”, ”Kategorisering” och ”Prioriteringssystem” och har inte implementerats ännu. Dessa påbörjade funktioner markeras med rött enligt tabell 5.1 och visar på en strävan efter att förbättra applikationens användbarhet genom att förenkla kommunikationen mellan arbetare och platschef. När dessa komponenter är fullständigt utvecklade kommer de sannolikt att förenkla samarbetet inom ett projekt eller dagliga aktiviteter. De funktioner som markeras med grått har endast en grundläggande funktionalitet för att visa kund konceptet, men hann ej bli färdigställt och behöver framtida utveckling.

Tabell 5.1: Tolkning av färgkod till figurer i kapitel 5.1.3

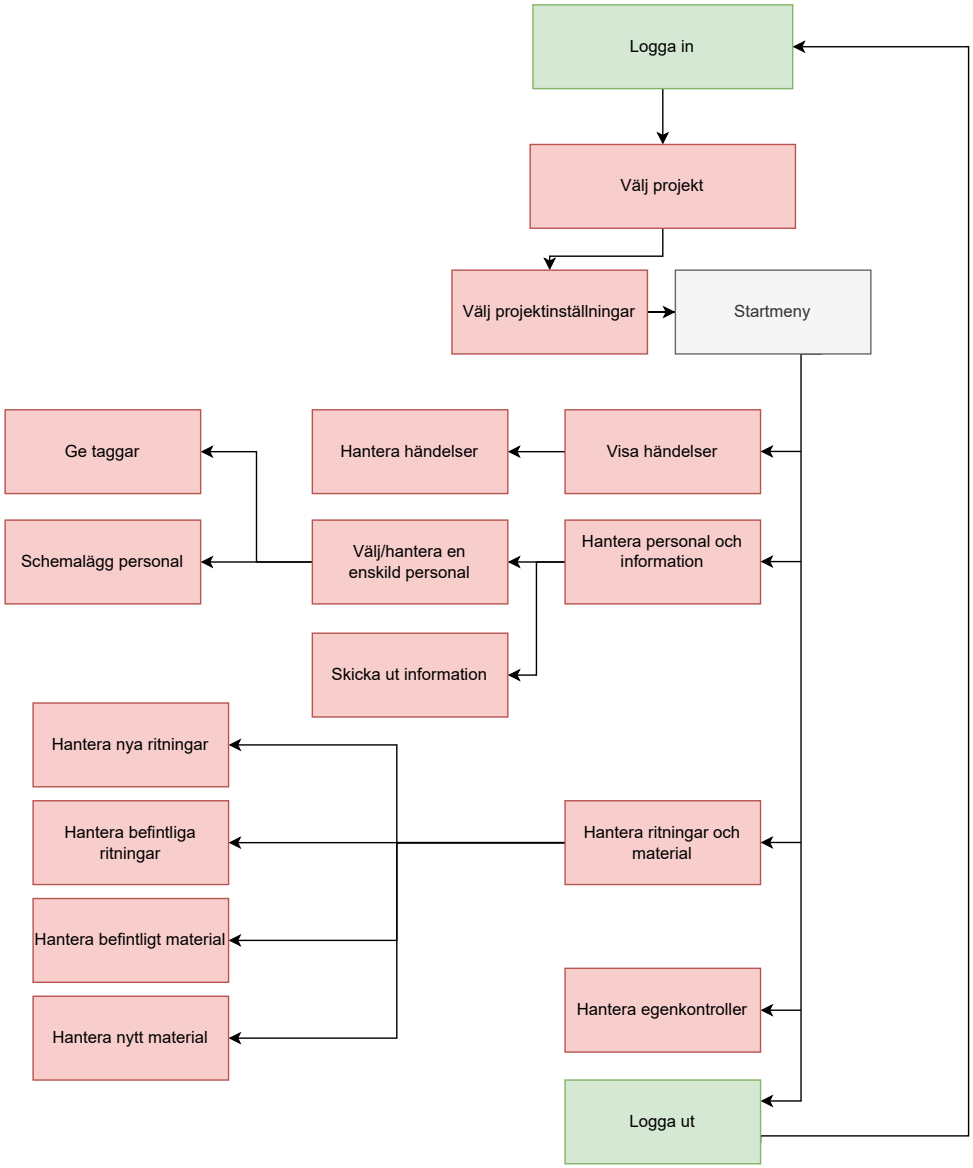
Färg	**Status**
Grön	Implementerad funktionalitet
Grå	Påbörjad funktionalitet
Röd	Ej påbörjad funktionalitet

Figur 5.1: Systemanatomin

För en överblick av den funktionella designen ges figur 5.2-5.3, dock blev alla block i trädstrukturerna inte nödvändigtvis implementerade till fullo. I menyträden kan alla block ses som olika fönster där pilarna visar hur sidorna är länkade tillsammans med knappar som förflyttar användaren mellan de olika fönstren.



Figur 5.2: Menyträd för arbetare



Figur 5.3: Menyträd för admin

5.2 Utvecklingsprocess

Detta kapitel beskriver de konkreta resultaten av utvecklingen av webbapplikationen, inklusive förstudie, kravhantering och kvalitetstester. Gruppen formulerade en detaljerad projektplan och utformade även en kravspecifikation se kapitel 1.5. En övergripande design för systemet och dess moduler fastställdes i form av en systemanatomin i början av projektet. Genom analys och diskussion etablerades därefter en passande systemarkitektur enligt riktlinjerna i gruppens arkitekturdokument. För att säkerställa kvaliteten på applikationen genomfördes tester för funktionalitet, prestanda och användarupplevelse. Testerna såg även till att kraven uppfylldes.

5.2.1 Förstudie

I början av projektet genomfördes en informationsinsamling av projektet främst genom flera kundmöten. Dessa möten var av betydelse för att identifiera och förstå de potentiella behoven hos byggarbetare. Dessa behov analyserades och formades till konkreta, mätbara och implementerbara krav. Sedan utforskades även olika tekniska aspekter vid detta stadiet.

Därefter inleddes diskussioner där varje medlem hade möjlighet att argumentera för vilka krav som skulle prioriteras vid implementationen av systemet. Målet med dessa diskussioner var att säkerställa en gemensam förståelse, värdering och enighet kring vilka krav som skulle prioriteras vid implementation. Genom denna process säkerställdes faktumet att arbetet endast var inriktat mot att uppfylla de mest väsentliga behoven hos kunden.

5.2.2 Arbetsmetoder

Under detta kapitel samlas projektgruppens resultat gällande arbetsmetoder.

5.2.2.1 Stand-up-möten

Projektgruppen hade dagliga stand-up-möten i ett standardformat som skickades i en Discordserver. Denna standardiserade rapportering tillförde en daglig översikt över individuella arbetsinsatser och fungerade för att motverka överlappningar av arbete. Med hjälp av dessa stand-up-möten hade gruppen strukturerad arbetskoordination vilket i sin tur bidrog till en ökad effektivitet och förebyggande av dubbelt arbete inom projektgruppen.

5.2.2.2 Scrum och Kanban

Att kombinera Scrum och Kanban fungerade som en metod för att inhämta erfarenhet genom att tillåta gruppen att kontinuerligt justera och förbättra sitt arbetssätt baserat på de utmaningar och insikter som uppstod under projektets gång. Denna metod gjorde det möjligt för gruppen att snabbt identifiera och implementera förbättringar, vilket resulterade i en dynamisk arbetsmiljö. Genom att ha flexibiliteten att integrera element från båda metoderna kunde gruppen bättre hantera oväntade hinder och effektivt möta projektets krav.

I slutändan möjliggjorde denna kombination av Scrum och Kanban en anpassningsbar och effektiv arbetsmetod som passade projektets förutsättningar. Den praktiska erfarenheten från att använda en hybridmetod bidrog till att öka gruppens kompetens och insikt i hur agila metoder kan anpassas för att maximera produktivitet och samarbete.

5.2.2.3 Parprogrammering

Projektgruppen använde parprogrammering konsekvent genom projektets gång där de 8 gruppmedlemmarna delades upp i 4 par. Utvecklingsledaren delade upp gruppmedlemmarna på bästa möjliga sätt så att expertisen delades upp jämnt över paren. Detta underlättade

för att få en jämn arbetsfördelning. Gruppmedlemmarna kunde inte alltid sitta i paren av olika anledningar men i största mån försökte paren att hålla ihop. Det observerades en påtaglig förbättring i kodkvalitet och effektivisering av problemlösning jämfört med när gruppen inte använde sig av parprogrammering. Metoden förbättrade inte bara gruppens tekniska förmåga utan stärkte också samarbetet och kommunikationen. Detta gjorde parprogrammering till en central del av arbetsprocessen.

5.2.2.4 Sprintar

Under projektets inledning skapade projektgruppen en strukturerad plan för att hantera arbetet genom att dela upp det i olika aktiviteter. En metod som gruppen valde att använda sig av var sprintar, inspirerat av Scrum, vilket skulle möjliggöra en tydlig uppdelning av arbetet och tillåta gruppen att fokusera på specifika delar av projektet under perioder på två veckor åt gången.

Tyvärr mötte projektgruppen utmaningar när det kom till att genomföra sprintarna effektivt. Trots ambitionen att följa sprintplaneringen exakt blev utförandet av sprintarna oregelbundet och svagt.

5.2.3 Kravprocess

Under genomförandet av projektet genomgick både kundens krav och gruppens krav förändringar, vilket direkt påverkade projektets utfall. Denna del av rapporten syftar till att detaljerat beskriva hur dessa krav anpassades och vilka effekter det hade på projektet.

Kravspecifikationen som utvecklades tillsammans med kunden var grundläggande för projektets framgång. Genom regelbundna möten och kontinuerlig feedback kunde kundens förväntningar omförhandlas. De som deltog främst på mötena var den analysansvariga och teamledaren, utöver kunden. Vid dessa möten visade gruppen upp en statusrapport av arbetet och antecknade feedback för att sedan anpassa produkten enligt kundens vision. Längre fram i projektet ledde kundens återkoppling till att signifikanta funktionella krav prioriterades över andra krav på grund av tidsbrist.

Vid utformningen av gruppens krav överestimerades effektiviteten med hänsyn till projektets fasta tidsram. Under projektets gång gjordes kontinuerliga reflektioner och revideringar av kraven, som anpassades både efter kundens önskemål och de givna tidsramarna. Genom en systematisk analys kunde gruppen specifikt dela in kundens krav i funktionella och icke-funktionella krav. Denna metodik säkerställde att de slutliga kraven inte bara återspeglade kundens förväntningar utan också var genomförbara inom projektets tidsmässiga begränsningar. Detta gjordes för att säkerställa leverans av en stabil och fungerande slutprodukt samtidigt som det lämnades utrymme för framtida implementeringar av återstående krav.

5.2.3.1 Påverkan på projektets framgång

Det kontinuerliga informationsutbytet med kunden och anpassningarna av kraven var avgörande för projektets framgång. Genom att integrera feedback som samlades in via kundmöten och anpassa arbetssätt utifrån det, lyckades projektgruppen utveckla en slutprodukt. Denna kunde presenteras och som uppfyllde kundens förväntningar. Denna metod bidrog till en effektiv hantering av de initiala och reviderade kraven, vilket i sin tur gav en stabil grund för projektets fortsatta framgång. Enligt gruppen och kunden ledde exempelvis anpassningarna till en förbättrad användarupplevelse. De ledde dessutom till ökad funktionalitet i slutprodukten, vilket speglar projektets lyckade förståelse och hantering av kundens behov och krav.

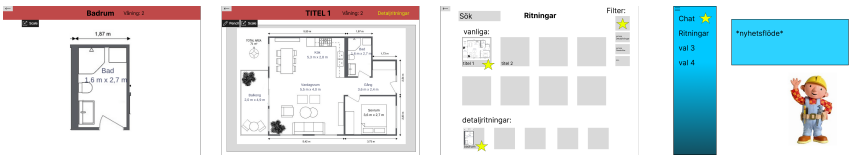
5.2.4 Testprocess

Under projektets gång genomgick testprocessen betydande förändringar. Eftersom att gruppen inte kunde bygga vidare på den tidigare kodbasen fanns mindre kod att testa än förväntat. För att minska på tiden som gick åt till tester och därmed lämna mer tid åt utveckling exkluderades därmed enhets- och integrationstester. Gruppen tog ett gemensamt beslut om detta då det ansågs att de tog upp för lång tid från projektet. Systemtester genomfördes dock som planerat med hjälp av Cypress, de användes för att säkerställa kvalitetskraven uppfylldes för projektet.

Automatiska tester i Lighthouse och Cypress sattes upp i början av projektet med hjälp av GitHub Actions. Projektgruppen testade flera olika konfigurationer, till en början testades hemsidan på flera operativsystem och webbläsare. Detta blev sedan ett problem när körtiden för tester blev begränsad i Github Actions och gruppen inte kunde köra fler tester. Detta ledde till att en egen server hos en av gruppens medlemmar fick användas för att kunna köra dessa automatiska tester. Utifrån detta konfigurerades testerna om för en mer lättsam testning som inte kördes på alla operativsystem och webbläsare. Lighthouse och Cypress kördes till slut för varje pull-förfrågan och varje push till grenarna `dev` och `main`.

5.2.5 Implementationsprocess

För att säkerställa effektiviteten i utvecklingsprocessen följdes alltid ett antal steg för att implementera ny funktionalitet i programmet. Först inleddes en diskussion med kunden och internt inom gruppen om hur den nya funktionaliteten skulle se ut. Därefter skapades en prototyp i designverktyget Figma, se figur 5.4. Prototypen presenterades för kunden som sedan avgjorde om funktionaliteten skulle implementeras eller inte. Om kunden godkände funktionaliteten placerades den på gruppens Kanban-tavla och arbetades igenom. Efter en viss utvecklingstid visades den nya funktionaliteten återigen för kunden som slutligen fick möjlighet att ge feedback på vad som var bra och vad som behövde förbättras.



Figur 5.4: Prototyp gjord i Figma

5.2.6 Versionshantering

I detta kapitel presenteras hur projektgruppen versionshanterade kod och dokument.

5.2.6.1 Versionshantering av kod

Projektgruppen använde sig av *feature branches* vid versionshantering i Github. Denna teknik innebar att nya funktioner utvecklades i separata grenar från huvudlinjen, vilket möjliggjorde isolerade tester och utveckling innan kod slogs samman med huvudgrenen. Denna praxis hjälpte till att hålla den primära kodgrenen stabil och säkerställde att endast genomtestad och verifierad kod integrerades, vilket bidrog till att upprätthålla projektets kvalitet och integritet.

5.2.6.2 Dokumenthantering

Till dokumenthantering använde projektgruppen Overleaf. Denna lösning möjliggjorde synkroniserat dokumentskrivande bland gruppmedlemmarna, vilket underlättade kommunikationen och minimerade fel i dokumenten.

5.2.7 Förbättring av process

I projektets initiala skede rådde viss förvirring och oklarhet inom gruppen. Gruppen hade till en början inte fullständig insyn i varandras arbetsuppgifter, vilket ledde till svårigheter i att koordinera arbetet. För att åtgärda dessa problem beslutades det att implementera dagliga stand-up-möten. Dessa möten hölls korta och gjorde det möjligt för gruppmedlemmarna att dela med sig av dagliga framsteg och utmaningar vilket bidrog till en förbättrad översikt och samordning. För att ytterligare stärka samverkan och säkerställa kontinuitet i både kodning och dokumentation introducerades också regelbundna lunchmöten mot slutet av projektet. Dessa möten erbjöd värde för att synkronisera arbetsprocesser och diskutera frågeställningar som uppstod under projektets gång. För att utvärdera dessa förbättringar genomfördes diskussioner på möten samt utvärdering via formulär där alla i gruppen kunde framföra sina åsikter muntligt och skriftligt. Genom dessa åtgärder kunde arbetsflödet effektiviseras och samarbetet förbättras inom gruppen.

5.3 Resultat av kraven

I kravspecifikationen fastställdes en serie krav som togs fram tillsammans med kunden. Dessa krav hade två olika prioriteringsnivåer, 1 och 2, där nivå 1 stod för grundläggande krav som skulle uppfyllas och nivå 2 som skulle genomföras i mån av tid. Kraven grupperades vidare in i tre kategorier: *kvalitetskrav*, *funktionella krav* och *icke-funktionella krav*. I det här kapitlet diskuteras i vilken utsträckning dessa krav har uppfyllts under projektets gång.

5.3.1 Kvalitetskrav

Kvalitetskraven har varit en väsentlig del i projektets utveckling för att säkerställa att produkten levererar den användarupplevelse och funktionalitet som önskades. Det första kvalitetskravet fokuserade på prestandan, specifikt att navigering mellan olika fönster inte ska ta mer än fem sekunder. För att säkerställa att detta krav uppfylldes genomfördes interna prestandatestningar under utvecklingens gång genom att mäta tiden med hjälp av en tidtagare, det genomfördes ett tiotals tester varje gång för att säkerställa tillförlitlighet. Testerna visade att navigeringstiden var under den gränsen, på bara några tiondels sekunder, vilket indikerar att webbapplikationen är snabb och effektiv med de nuvarande funktionerna som finns implementerade.

Det andra kvalitetskravet var att webbapplikationen ska vara responsiv. Genom att använda React och TailwindCSS implementerades en flexibel och responsiv design. Reacts modulära komponentstruktur i kombination med Tailwinds omfattande bibliotek underlättade att skapa ett användargränssnitt som är responsiv och dessutom skalbar. Användargränssnittet anpassar sig efter olika enheters skärmstorlekar. Detta testades genom att projektmedlemmarna testkörde webbapplikationen på respektive persondator med olika skärmstorlekar.

Det tredje kvalitetskravet rörde webbapplikationens användarvänlighet, med ett fokus på navigationsdjupet som inte skulle vara mer än åtta. Detta krav uppfylldes då den nuvarande designen har ett maximalt djup på tre.

5.3.2 Funktionella krav

När kravspecifikationen utformades definierades flera funktionella krav. Under projektets gång har några nyckelfunktioner som ansetts vara väsentliga för att förbättra användarinteraktionen och effektivisera arbetet på byggarbetsplatser implementerats. Två av dessa är bland annat inloggningsprocessen och hantering av ritningar. För närvarande hanteras inloggningen med testdata, kallad ”dummy-data”, för att åskådliggöra och testa de nuvarande funktionerna.

Efter inloggning presenteras för användaren en startsida som är anpassad efter hens geografiska plats för att ta fram det lokala vädret och hen får en översikt över senast besökta, favoritmarkerade och nyligen ändrade ritningar. Användarens interaktion med dessa ritningar är dock begränsad då viss funktionalitet kräver vidare utveckling och back-end funktionalitet saknas helt.

Från startsidan kan användaren navigera via sidomenyn till olika funktioner som ”Välj ritning”, ”Information”, ”Frågor och svar”, samt ”Logga ut”. Användaren kan i nuläget interagera med en ritning och öppna den i fullskärmsläge. I framtiden förväntas applikationen låta användaren välja en specifik byggnad, våning och sektion för att filtrera ut en önskad ritning. När en ritning har öppnats finns en möjlighet att se relaterade detaljritningar i en lista där möjligheten finns att svepa alternativen via en pop-up.

I framtiden skall applikationen ha ett interaktivt system för frågor och svar där användare kan skapa frågor relaterade till specifika ritningar utan också se händelser på tidigare skickade ärenden. Genom att välja en kombination av alternativen att ”markera på en ritning”, ”ta en bild” eller ”skriva ett meddelande” får användaren upp en pop-up för vidare åtgärder. Detta medför att en snabb och smidig dialog kan påbörjas med projektledare eller andra överordnade. Dessa funktioner illustrerar systemets kapacitet att anpassa sig till användarnas behov och effektivisera arbetsprocesser.

5.3.3 Icke-funktionella krav

I kravspecifikationen fastställdes två icke-funktionella krav för att garantera ett system som inte bara uppfyller de funktionella behoven utan även levererar en produkt som ger en tillfredsställande användarupplevelse, oberoende av användarens tekniska erfarenhet. Dessa krav, med nivå 2 prioritering, var att ”skapa estetiskt tilltalande menyer och knappar” samt att ”inkludera grundläggande flerspråkigt stöd”. Trots deras lägre prioritering uppfylldes båda kraven.

Med tanke på att många användare inom byggbranschen inte nödvändigtvis har stor teknisk erfarenhet var det av stor vikt att utforma ett användargränssnitt som är både enkelt och intuitivt. Med hänsyn till detta utvecklades det nuvarande systemet med ett minimerat navigationsdjup och fokus lades på tydliga och lättförståeliga navigationsknappar. Detta underlättar för användarna att snabbt och effektivt hitta det som behövs.

Det andra kravet att inkludera grundläggande stöd för flerspråkighet var också avgörande för att tillgodose en arbetsstyrka som är flerspråkig. Inom byggbranschen är det vanligt att personal har olika språkbakgrunder vilket gör att detta icke-funktionella krav är av vikt för att erbjuda en produkt som kan användas enkelt och smidigt. I dagsläget har front-end-utvecklingen lagt en grund för detta stöd.

5.4 Inverkan på byggindustri

Projektet har potential att revolutionera byggindustrin genom att effektivisera kommunikationen och hanteringen av material och ritningar på byggarbetsplatser. Om följande utökade funktioner implementeras kommer projektet att leda till en mer effektiv och säker byggindustri, vilket sparar både tid och pengar och främjar smidigare arbetsflöden och bättre resursutnyttjande.

Kommunikation är ett område där förbättringar kommer att kunna ses. Genom att implementera en plattform där byggarbetare kan rapportera fel och ställa frågor direkt till chefer, arkitekter och andra ansvariga ökar effektiviteten för kommunikation. Byggarbetare behöver

ofta ställa frågor till chefen, som vid behov även tar vidare frågan till rätt person. När frågan behöver tas vidare kan viktiga detaljer gå förlorade och det tar ofta lång tid att få svar. Stora mängder tid spenderas även på att söka efter de ansvariga på platsen. Applikationens system för att skicka in frågor löser samtliga av dessa problem, vilket sparar värdefull tid och gör det möjligt för arbetarna att fokusera mer på själva byggprocessen. Funktionaliteten för administratörer att vidarebefordra en fråga leder också till att ingen information går förlorad då en fråga behöver tas vidare, eftersom frågan kommer fram i sin originalform.

En integrerad GPS-funktion för spårning av material löser ett vanligt problem med att material ofta flyttas utan att platsen finns dokumenterad. Under vinterhalvåret när materialet även blir täckt av snö försvåras arbetet ytterligare. Genom att utrusta material med GPS-sändare kan personalen med hjälp av applikationen enkelt lokalisera och hämta det nödvändiga materialet, utan att behöva spendera tid på att leta genom vad som vanligtvis är väldigt stora områden.

Applikationen kommer att bidra med en ökad transparens under byggprojekt. Genom att få all information dokumenterad i systemet till exempel bygghandlingar, inköpsinformation och om transporter kommer i tid, går det att som beställare få mer insikt i vart pengarna går. Versionshantering av ritningar erbjuder också ökad transparens genom möjligheten att spåra förändringar över tid, vilket underlättar för alla inblandade att hålla sig uppdaterade och säkerställa att de arbetar med korrekta och aktuella uppgifter.

Säkerhet är även ett viktigt fokusområde av flera anledningar. Med begränsad åtkomst till ritningar finns det möjlighet att skydda känslig information och säkerställa att bara behörig personal har tillgång till dem. Det går att stänga av åtkomst till alla dokument för vissa områden, till exempel om en våning saknar säkerhetsräcken, för att byggarbetarna inte ska arbeta på platser som inte är säkra. Ett annat problem som finns idag är att dokument som sällan används skrivs ut i en enda papperskopia som förvaras i chefens kontor. Detta leder ofta till improvisationer från arbetarnas håll vid de tillfällen chefen inte finns lättillgänglig, improvisationer som ibland äventyrar säkerheten på byggarbetsplatsen. Genom att dokumenten finns lagrade digitalt har arbetarna alltid tillgång till dessa, och kan därmed enkelt leta upp de säkerhetsinstruktioner som finns.

5.5 Implementationsvärde

Under utvecklingen av projektet ställdes frågan om hur webbapplikationen kunde implementeras för att skapa värde för kunden. Trots ansträngningar och den planering som genomfördes uppkom hinder som begränsade möjligheten att implementera alla krav enligt den ursprungliga kravspecifikationen. Dessa begränsningar innefattade tidsbegränsningar samt komplexiteten för vissa funktioner. Det är viktigt att notera att trots hindren lyckades webbapplikationen skapa värde för kunden genom de funktioner som implementerades.

Webbapplikationen som har skapats under projektets gång är en grund för att optimera hanteringen av ritningar och förbättra kommunikationen inom byggbranschen. Genom att webbapplikationen möjliggör lagring av ritningar och underlättar kommunikationen på detta vis adresseras de utmaningarna som byggbranschen står inför - bristfällig kommunikation och felhantering av ritningar.

En av de primära fördelarna med webbapplikationen är den snabba tillgången till uppdateringar för både ritningar och meddelanden, via nyhetsflödet, från överordnade angående olika ärenden. Genom att ge användarna möjligheten att snabbt identifiera och reagera på förändringar när de loggar in, har arbetsflödet effektiviserats och risken för missförstånd minskats.

En annan nyttig funktion är möjligheten för användarna att tydligt identifiera vilken version av en ritning som används. Genom att tillhandahålla denna funktion kan användarna enkelt hålla koll på ritningshistoriken och säkerställa att de arbetar med den senaste versionen, vilket ökar noggrannheten och effektiviteten i deras arbete.

Designen av webbapplikationen är skräddarsydd efter kundens behov och fokuserar på enkel navigering och användarvänlighet. Den är responsiv och kompatibel med olika enheter, vilket ger användarna en smidig upplevelse oavsett vilken storlek det är på enheten de använder. Dessutom är koden modulär och komponentbaserad, vilket gör det enkelt att införa nya funktioner i framtiden och möjliggör kundens önskan om att utöka projektet med nya funktioner.

Sammanfattningsvis har ansträngningarna att implementera dessa mervärdesfunktioner, trots de utmaningar som uppstod, resulterat i en webbapplikation som effektiviserar arbetsprocessen, ökar noggrannheten och förbättrar kommunikationen för användare inom byggbranschen.

5.6 Gemensamma erfarenheter

I detta avsnitt genomförs en noggrann reflektion över de processrelaterade och tekniska insikter som gruppen har ackumulerat under projektets gång. Vikten av effektiv kommunikation inom gruppen och tillämpningen av Scrum som arbetsmetod framhävs. Vidare diskuteras det tekniska övergången från Vue till React, fördjupningen i TypeScript, den komponentbaserade arkitekturen med React, integrationen av Next.js, genomförandet av automatiska tester i GitHub och hur tillämpningen av *runner*-tid kan påverka projektet.

5.6.1 Processrelaterade erfarenheter

I detta kapitel presenteras processrelaterade erfarenheter.

- Gruppens val av Discord möjliggjorde en oavbruten och omedelbar dialog mellan gruppmedlemmarna, även under perioder av distansarbete. Discord har inte bara fungerat som en plattform för textmeddelanden, utan har även tillhandahållit möjligheter för röst- och videosamtal. Detta förbättrade i sin tur kapaciteten att effektivt adressera och lösa komplexa problem, samtidigt som det bidrog till att upprätthålla en stark känsla av samhörighet inom gruppen.
- Gruppens regelbundna möten, inklusive handledarmöten, lunchmöten och dagliga stand-up-möten, utgjorde en väsentlig del i resultatet för projektet. Dessa sammanträden tillhandahöll en stabil struktur för planering, uppgiftsdelegering och synkronisering av individuella utföranden. Genom att ständigt involvera alla gruppmedlemmar i dessa möten kunde en form av garanti lämnas där alla var uppdaterade och inriktade på projektets mål. Detta bidrog i sin tur till att upprätthålla en god effektivitetsnivå genom hela projektets livscykel.
- Beslutet att övergå från Vue till React medförde att den befintliga kodbasen behövde överges och därefter påbörjades en ny kodbas. Detta utgjorde initialt en utmaning, men resulterade slutligen i betydande fördelar. Reacts komponentbaserade arkitektur, tillsammans med dess omfattande ekosystem av verktyg, inklusive Next.js, tillhandahöll gruppen med ökad flexibilitet och förbättrad prestanda för projektet. Denna tekniska övergång krävde dock en viss inlärningskurva och anpassning från gruppens sida. Att byta till React var inte ett självklart beslut, men eftersom kunden efterfrågade en ny kodbas i React gjorde vi omställningen. React komponentbaserade arkitektur och rika

ekosystem möjligjorde såväl modulär som anpassningsbar kod. I och med att syftet var just en modulär kod samt hög anpassningsbarhet, eftersom projeketet hade många utvecklare i en och samma kodbas, var React ett bra alternativ. Till framtiden är React bra för liknande projekt.

- Projektet använde GitHub flitigt, särskilt för tillämpning av grenar för olika funktioner, vilket var avgörande för effektiv hantering av kodändringar och för att säkerställa stabiliteten i huvudkodbasen. Detta system möjliggjorde experiment och utveckling av nya funktioner utan att störa den pågående utvecklingen, vilket i sin tur minimerade risken för kodkonflikter.
- För att säkerställa att alla gruppmedlemmar var kontinuerligt informerade om projektets framsteg och de individuella ansvarsområdena, implementerades en Kanban-tavla direkt i GitHub. Detta verktyg bistod gruppen i visualiseringen av arbetsflödet och övervakningen av framstegen för varje uppgift i realtid. Detta underlättade för gruppmedlemmarna att prioritera arbetsuppgifter och hålla sig uppdaterade om projektets status.
- I ett initialt skede av projektet visade sig arbetsfördelningen vara ineffektiv på grund av gruppens begränsade insikt i projektets detaljer. För att adressera detta genomfördes en grundlig analys av projektets behov, vilket möjliggjorde en bättre strukturering av arbetsuppgifterna. Genom att justera ansvarsområden och stärka intern kommunikation, lyckades gruppen förbättra effektiviteten i arbetsfördelningen, vilket i sin tur snabbade på utvecklingsprocessen.
- Att fatta arkitektoniska beslut i Vue, följt av en övergång till React, var en betydande lärdom för projektgruppen. Ursprungligen valdes Vue för dess förmåga att effektivt hantera dynamiska användargränssnitt, vilket stödde de initiala projektmålen väl. När projektets omfattning och kundens krav förändrades blev React däremot mer fördelaktigt, speciellt med tanke på dess stöd för större skalbarhet och flexibilitet i projektutvecklingen. Det var inte ett självklart beslut att byta till React, men eftersom kunden önskade det gjorde vi anpassningen. I och med att webbapplikationen skulle vara modulär och projektet skulle vara relativt stort, med många utvecklare inblandade, passade React bra för denna uppgift. Denna erfarenhet innebar en dubbel utmaning: dels att omstrukturera den befintliga arkitekturen från Vue till React, dels att anpassa gruppens tekniska kunskaper till en ny ramverksstruktur. Trots den inledande inlärningskurvan och att det initialt saknades djupare kunskaper i ramverket, bidrog övergången till att stärka gruppens förmåga att snabbt anpassa sig till förändrade tekniska krav. Till framtida projekt är en viktig lärdom att grundligt undersöka bästa ramverk och programmeringsspråk i förhållande till storlek, komplexitet och modularitet av uppgiften.
- Aktivt lyssnande gentemot kunden framhävdes som en grundläggande aspekt av kommunikationen och samarbetet under projektets gång. Denna förmåga var avgörande för att noggrant förstå och tolka kundens specifika behov och önskemål. Genom införandet av regelbundna och strukturerade kundmöten säkerställdes att varje gruppmedlem aktivt lyssnade, vilket resulterade i mer precisa och detaljerade kravspecifikationer. Denna praxis inte bara fördjupade förståelsen för projektets mål, utan stärkte också relationen med kunden genom att visa på ett genuint intresse och engagemang för att leverera en produkt som väl motsvarade förväntningarna.

5.6.2 Tekniska erfarenheter

I detta kapitel presenteras tekniska erfarenheter.

- Applikationen skapades först som en multi page application, men omarbetades senare till en single page application. Genom att ladda innehåll dynamiskt utan att behöva ladda om hela sidan skapades en känsla av kontinuitet och ökad responsivitet vid byte mellan olika sidor. De komponenter som används behöver dessutom enbart laddas in en gång, vilket minskar tiden för att ladda in en ny sida.
- Gruppen valde att använda TypeScript, React och Next.js för projektet baserat på kundens önskan att omstrukturera det befintliga systemet med React. Valet motiverades av erfarenhet inom gruppen och fördelarna med varje teknik: TypeScripts förbättring av kodstabilitet, Reacts komponentbaserade arkitektur och Next.js för dess stöd för serverrendering och routing. Denna uppsättning av tekniska verktyg möjliggjorde en effektiv utvecklingsprocess och levererade en modern och prestandaoptimerad webbapplikation.
- Gruppen valde att dela upp projektet i generiska komponenter för att kunna återanvända komponenter och få en enhetlig design. Genom att skapa komponenter som kan återanvändas över hela webbplatsen minskar utvecklingstiden och underhåll av koden underlättas.
- Användningen av TailwindCSS för designen ger en strukturerad och konsekvent metod för att designa komponenter. Gruppen kompletterade TailwindCSS med lite egenskriven CSS för att hantera specialanpassade designbehov eller komplexa interaktioner som inte täcks av standardkomponenterna. Denna kombination av TailwindCSS och anpassad CSS möjliggjorde en effektiv utvecklingsprocess och resulterade i en enhetlig, stilren och lättunderhållen webbplats.
- Gruppen valde att använda automatiska tester för att testa koden som skrevs i projektet. I början lades en bra plan ut för att kunna täcka så mycket som möjligt. Detta gick dock inte helt som planerat. Då GitHub begränsar testers körtid fick gruppen snabbt slut på körtid. En alternativ lösning hittades som involverade att minska antalet tester och att köra dem på en egen server istället för hos GitHub.
- Gruppen hade som mål att hemsidan skulle vara dynamisk och fungera bra på både surfplattor för byggarbetare och på datorer för chefer. Gruppen fick lära sig mycket och använda projektets verktyg på bästa sätt.
- Gruppen valde att använda Microsoft Excel för att skapa en omfattande uppgiftslista. Dessa klart definierade ansvarsområden för varje teammedlem användes för att organisera arbetsflödet. Microsoft Excels funktioner som pivot tabeller, filter, sortering och olika formler utnyttjades för att gruppen skulle kunna skapa ett dynamiskt och optimerat arbetsflöde.

6 Diskussion

I detta avsnitt kommer föregående avsnitt 5 att förklaras i djupare detalj. Resultatet och metoden kommer att diskuteras i ett sammanhang där de värderas. Gruppen kommer dessutom att tydliggöra sin tolkning av resultatet och tankar kring projektets framtid. Dessutom kommer en hållbarhetsanalys att göras där ett källkritikiskt tänkande tillämpas.

6.1 Analys av resultat

Projektets strategiska beslut att endast hantera front-end-utveckling och inte lägga någon vikt på back-end-funktionaliteterna är begripligt med tanke på tidsbegränsningen på 400 timmar per person. Denna avgränsning har dock inneburit att viktiga funktioner som versionshantering av ritningar och en effektiv överföring av ritningsfiler inte kunnat implementeras. Denna brist kommer i längden att begränsa systemets effektivitet och skalbarhet. Detta kommer att behöva implementeras vidare i framtiden.

Den potentiella inverkan av ett välutvecklat digitalt ritningshanteringsprogram på byggindustrin är betydande, men då omstrukturering av en befintlig produkt genomfördes var det dessvärre inte möjligt att uppfylla alla krav. Trots de nuvarande begränsningarna i projektets första version lägger den en stabil grund för framtida utvecklingar och erbjuder en plattform från vilken ytterligare funktioner och förbättringar kan utforskas och implementeras.

6.1.1 Fortsättning från tidigare projektgrupp

Att fortsätta med ett projekt som har överlämnats från en tidigare projektgrupp kan vara både en fördel och en nackdel. I det här fallet hade den tidigare gruppen utvecklat en hemsida i VueJs, men kunden var inte helt nöjd med resultatet och ville att koden skulle ses över. Därmed togs beslutet att skriva om hela kodbasen i React utifrån en rekommendation från en konsult som hade sett över projektet. Projektgruppen var då tvungen att genomgå ytterligare utbildning för detta. Detta resulterade i en överflödig tidsåtgång för utbildningsändamål, vilket i sin tur genererade en betydande mängd frustration inom gruppen.

Detta blev en lärdom för gruppen, att tydlig kommunikation och planering från första början är viktigt för framgången hos projektet, särskilt när det gäller teknikval och övergångar mellan olika verktyg eller ramverk. En mer effektiv strategi i detta fall hade varit att från början bestämma att projektet skulle migreras till React och sedan anpassa resurser och utbildning därefter. Fördröjningen, vilken uppstod till följd av nödvändigheten att hantera två distinkta teknologier vid projektets begynnelse, kunde potentiellt ha reducerats.

Det viktiga är att dra lärdomar från erfarenheter och att implementera bättre kommunikation, samt planering för att undvika liknande situationer i framtiden. Detta kan bland annat göras genom att införa stand-up-möten, att vara proaktiv och strategisk i teknikval samt övergångar mellan olika ramverk. Genom att göra detta kan både tid och resurser sparas, dessutom kan det bidra till ett smidigare och mer framgångsrikt projektgenomförande.

6.1.2 Värdering av lösning

Från resultaten framgår det att projektet har tagit viktiga steg mot att skapa en grundläggande men funktionell applikation trots signifikanta begränsningar. Det viktigaste projektgruppen har skapat är ett användarvänligt GUI-system som är optimerat för touch-baserade enheter. Detta är av stor betydelse då användare i byggindustrin lättare förstår sig på och föredrar att använda sina fingrar än en datormus och tangentbord. Därmed har webbplatsen utformats för att vara intuitiv och lättanvänd även för de med begränsad teknisk kompetens, vilket är av direkt värde för användaren då det underlättar inlärningsprocessen. Utöver det finns fördelen att operativsystem för surfplattor ofta hanterar inzoomningen på skärmen vilket gör att användare kan zooma in på en ritning.

Några nackdelar med projektgruppens lösning var att kravspecifikationen blev för stor då resultatet blev att endast hälften av kraven med högsta prioritet blev avklarade. Detta skedde främst på grund utav att mycket tid gick åt annat än kodande, exempelvis dokumentskrivande som tog upp mycket tid från projektet. Detta är negativt både för kunden som blev lovad mer än vad som levererades och för gruppmedlemmarna som blev stressade mot slutet av projektet då insikten att alla krav ej skulle komma att fullbordas sjönk in. Utöver det finns det säkerligen många brister i implementationen då gruppen helt enkelt inte var bekanta med arbetsuppgifterna.

Detta är något som kan tyckas vara oundvikligt då gruppmedlemmarna inte kunde avgöra om vare sig en implementation var bra eller dålig, det enda som var observerbart var om koden var funktionell eller inte. Dock hade detta lösts med en bättre och mer strukturerad planering av tiden genom att be en utomstående part att se över koden och få feedback om den var lättläslig och begriplig utöver funktionell. I det här fallet kommer en utomstående konsult att granska koden och ge feedback från kundens håll men för gruppen är detta en erfarenhet som alla bär med sig till andra projekt framöver.

6.1.3 Alternativa implementationssätt

Under utvecklingen av detta projekt övervägde vi flera alternativa lösningar för att implementera webbapplikationen. Det som användes av tidigare grupper som arbetade med detta projekt var Vue.js. Dock använde dessa grupper en version som inte längre hade stöd. När vår grupp påbörjade projektet fanns det stöd för en nyare version, vilket skulle ha inneburit att vi behövde skriva om koden till den nyare versionen. Enligt kundens externa konsult var den tidigare gruppens kod inte välskriven och implementationen kunde ha utförts på ett mer effektivt sätt genom att välja något annat än Vue.js.

En annan viktig aspekt som diskuterades var hur hemsidan skulle renderas – på servern eller på klienten. För närvarande är hemsidan implementerad i Next.js, vilket stödde serverrendering. Trots detta användes serverrendering inte alltid i det nuvarande projektet, och mycket av innehållet renderades på klienten. Detta ledde till frågan om det skulle ha varit fördelaktigt att öka användningen av serverrendering istället för klientrendering, och vilka potentiella för- och nackdelar detta skulle ha inneburit för prestanda och användarupplevelse.

Ett centralt beslut under projektets gång var att följa konsultens rekommendationer för teknologival och implementationssätt. Ett alternativ till detta hade varit att genomföra en egen utredning för att utforska och jämföra olika teknologier och renderingsmetoder mer ingående. En sådan utredning kunde ha inneburit prestandatester, användarstudier och analyser av hur olika implementationssätt skulle ha påverkat utvecklingen av applikationen.

Genom att noggrant överväga dessa alternativa lösningar skulle vi ha kunnat optimera projektet för att bättre möta de specifika behoven och målen. En djupare utredning hade möjligen kunnat ge en mer omfattande förståelse av de olika alternativen och deras långsiktiga implikationer, vilket skulle ha kunnat bidra till en mer robust och framtidssäker arkitektur.

I framtida projekt kan det vara värt att överväga en sådan utredning för att säkerställa att alla möjliga implementationssätt är utforskade och att det valda tillvägagångssättet är det mest lämpliga baserat på en bredare analys.

6.1.4 Fortsatt utveckling av applikationen

Erfarenheter från detta projekt kan ge värdefulla lärdomar för framtida utvecklingar av en linkande applikation. Vikten av att tydligt definiera och kommunicera projektets omfattning och begränsningar är en sådan lärdom. Detta kan hjälpa till att sätta realistiska förväntningar och fördela resurser mer effektivt. Ytterligare en lärdom är betydelsen av att utveckla en robust systemanatomi tidigt i projektet. Genom att detaljerat kartlägga systemets komponenter och deras interaktioner kan projektgruppen identifiera och behandla potentiella problem innan de uppstår, vilket minimerar riskerna och förbättrar systemets övergripande prestanda och stabilitet.

För att fortsätta utvecklingen och fullbordandet av detta projekt krävs det att de återstående komponenterna som inte blev implementerade ses över. Detta bör göras via ett kundmöte för att säkerställa att applikationen utvecklas enligt kundens vision och blir en ännu mer omfattande verktygssats som stöder sina användare.

För att få en applikation med en rik uppsättning funktioner som dessutom är anpassade för en effektiv och dynamisk arbetsmiljö krävs det kännedom om målgruppen av användare. Med kännedom om målgruppen och design av användarvänliga komponenter ges en stark grund till resten av projektet. Lärdomen som bör tas med från projektet är att inte överskatta vad som kan hinnas med och skapa en tydlig projektplan som alla kan följa. En fördel är att planera för att framtida expansioner och förbättringar kommer att göras av andra individer än gruppen själv och försöka att möta kunden regelbundet för att bygga tillit.

6.2 Utförandet av projektet

I detta avsnitt diskuteras metoden för utförandet av projektet. Dessutom tas regelbundna processer som användes upp, se avsnitt 4.

6.2.1 Erfarenhetsprocessen

Under projektets gång har erfarenhetsutbytet och kunskapsöverföringen inom gruppen varit avgörande för att navigera genom tekniska och processrelaterade utmaningar. Regelbundna reflektionsmöten bidrog till att identifiera effektiva strategier och områden för förbättring. Ett framträdande exempel på detta är övergången från VueJs till React, där gruppen genom gemensamma problemlösningsessioner kunde anpassa sig till den nya tekniken, vilket resulterade i en smidigare utvecklingsprocess och förbättrad kodkvalitet.

Användningen av Scrum som arbetsmetod möjliggjorde för gruppen att snabbt anpassa sig till förändrade projektbehov och individuella medlemmars kompetenser. Genom dagliga stand-up-möten och veckovis sprint-retrospektiv kunde arbetsflöden justeras dynamiskt, vilket ökade effektiviteten och stärkte samarbetet inom gruppen. Denna erfarenhetsprocess, där gruppen kontinuerligt lärt från varje projektsteg, har inte bara stärkt den tekniska förmågan utan också gruppens förmåga att agera agilt och responsivt inför nya utmaningar.

6.2.1.1 Tidsrapportering

Tidigare i avsnitt 2.2.2 och 4.1.1 nämns tidsrapportering för projektet. *Google Sheets* enkelhet och gruppens bekantskap med verktyget möjliggjorde en smidig initiering av tidsrapporteringen och gav en god översikt över arbetsinsatserna. Genom att logga arbetstimmar i detaljerade intervall kunde varje gruppmedlem enkelt se hur mycket tid som spenderats och hur mycket mer som behövde läggas ner.

Denna metod stödde en jämn fördelning av arbetsbelastningen genom att tydligt visa vilka uppgifter som krävde mer tid och vilka som kunde delegeras för att balansera arbetsbördan. Dessutom tillät den kontinuerliga uppföljningen av projekttiden oss att identifiera potentiella förseningar tidigt och att justera våra tidsplaner för att undvika förseningar. Detta bidrog till att hålla projektet enligt tidsplanen genom att vi kunde reagera proaktivt på problem och omfördela resurser effektivt. Detta minskade även risken för överbelastning hos enskilda teammedlemmar utan att lägga till onödig komplexitet i arbetsprocessen.

6.2.1.2 Stand-up-möten

Implementeringen av dagliga stand-up-möten och regelbundna lunchmöten har spelat en stor roll för projektet. Gruppens initiala utmaningar med förvirring och bristande översikt över varandras arbetsuppgifter tydliggjorde behovet av bättre kommunikationsstrukturer. Dessa möten har inte bara underlättat snabbare problemidentifiering och lösning utan även bidragit till en starkare sammanhållning inom gruppen.

Vidare är det viktigt att diskutera hurvida de implementerade mötena var den optimala lösningen eller om det finns andra strategier som potentiellt kunde ha varit mer effektiva. Det finns en möjlighet att utforska hur dessa möten kunde anpassas för att bli ännu mer effektiva, exempelvis genom att införa tydligare agendor eller genom att förändra frekvensen baserat på projektets fas.

6.2.1.3 Scrum och Kanban

Projektgruppen integrerade olika aspekter av både Scrum och Kanban för att organisera arbetet. Från Scrum implementerades stand-up-möten för daglig samordning och sprintar, vilka planerades med en tvåveckorsperiod av utvecklingsledaren. Under varje sprint arbetade programmeringspar tillsammans för att slutföra de tilldelade uppgifterna inom sprintens tidsram.

Från Kanban användes principen med en Kanban-tavla för att ge struktur åt arbetsflödet. En utmaning som gruppen stötte på var att Kanban-tavlan var integrerad i GitHub, vilket begränsade dess flexibilitet för att hantera icke-kodrelaterade aktiviteter, till exempel dokumentation. Det blev delvis svårt att organisera sådana aktiviteter effektivt eftersom Kanban-tavlan var fokuserad på koden. Att använda en separat tjänst för Kanban-tavlan för icke-kodrelaterade aktiviteter skulle ha varit fördelaktigt för att öka organisationen och tydligheten för projektgruppen.

Trots utmaningarna med att integrera Kanban-tavlan i GitHub, erbjöd den ändå struktur åt arbetsflödet och underlättade visualiseringen av pågående och avslutade uppgifter. Dock var en av utmaningarna att inte alla gruppmedlemmar var konsekventa med att uppdatera tavlan regelbundet, vilket påverkade dess effektivitet som ett verktyg för arbetsstyrning och samordning.

Sammanfattningsvis kombinerade projektgruppen element från både Scrum och Kanban för att organisera arbetet. Att använda stand-up-möten, sprintar och Kanban-tavlan bidrog till strukturerade arbetsflöden och förbättrad samordning inom gruppen, även om det fanns utmaningar med att integrera och upprätthålla dessa system effektivt.

6.2.2 Kravprocessen

Projektet inleddes med ett initialt möte med kunden och hela gruppen, där kunden fick möjlighet att presentera och förklara vad projektet innebar, vad behoven var och vilka önskemål som fanns. Kunden visade även upp ett flödesschema som hade utvecklats med åren för hur tanken är att allt ska fungera. Det var allt ifrån knapparnas funktionalitet till vilken information som ska vara tillgänglig. Syftet var att skapa en produkt som var användarvänlig och lättförståelig, anpassad för användare utan djupgående teknisk erfarenhet. För gruppen som är van med teknik och digitala produkter, krävdes det att skapa en design som var anpassad för en mer grundläggande nivå.

Efter det första mötet hölls många interna diskussioner för att konkretisera kundens vision till en teknisk plan. I denna del av kravprocessen hölls ett flertal möten för att omvandla diskussioner till faktiska krav. Utöver interna gruppmöten hölls även regelbundna möten med handledaren där det diskuterades om de utformade kraven var genomförbara och om gruppen höll sig på en realistisk nivå i förhållande till kursens tidsram.

Gruppen hade även ytterligare kundmöten för att säkerställa att de tekniska krav som utformats var i enlighet med kundens förväntningar. Kravspecifikationen kunde utformas, som i sin tur delades med kunden för granskning och återkoppling. Det var en viktig del i kravprocessen då det här framkom om det fanns en gemensam förståelse för projektets mål och om kundens önskan och behov tolkats korrekt.

Då kraven var många och det fanns en tidsram att förhålla sig till delades kraven upp i två prioriteringsnivåer, nivå 1 och nivå 2. De krav som var markerade med nivå 1 var de väsentliga kraven och målet var att lyckas implementera dem under projektets gång. Kraven med lägre prioritet var snarare sådant som skulle implementeras om tid fanns. För att effektivisera utvecklingen och hanteringen av kraven delades även kraven upp i olika kategorier; kvalitetskrav, funktionella krav och icke-funktionella krav.

När kunden bekräftat att kravspecifikationen var korrekt tolkad och godkänd gick gruppen vidare med implementering och utveckling. Genom att gå igenom en kravprocess på detta vis kunde det säkerställas att slutprodukten, så långt som gruppen lyckades utveckla den, kunde uppfylla både kursens mål och kundens behov. Kravprocessen i detta projekt har in-

te bara förstärkt vikten av god kommunikation utan även belyst vikten av flexibilitet och anpassningsbarhet, något som gruppen lyckades med exceptionellt.

6.2.2.1 Kvalitetskrav

Kvalitetskraven var centrala för att utveckla en produkt som inte bara uppfyllde funktionella behov utan också var användarvänlig. Det första kvalitetskravet, som fokuserade på prestanda, specificerade att navigeringen mellan olika flikar inte skulle överstiga fem sekunder. För att validera detta genomfördes prestandatestningar, vilka visade att navigeringstiden var under den uppsatta gränsen på fem sekunder. Resultatet visade på några tiondels sekunders navigeringstid.

Trots dessa positiva resultat identifierades flera områden för potentiella förbättringar. Testerna begränsades till persondatorer och tog inte hänsyn till surfplattor, som produkten också ska användas på av byggarbetarna ute på arbetsplatsen. Surfplattor kan ha andra prestandaegenskaper, som svagare processorer eller mindre skärmar, som kan påverka laddningstider och responsivitet. Dessutom utfördes testerna under ideala förhållanden utan att simulera hög belastning eller nätverksfördröjningar, vilket inte riktigt speglar de miljöer där produkten främst kommer att användas. För framtida projekt är det avgörande att inkludera dessa faktorer i testplanen. Genom att till exempel genomföra stresstester eller belastningstester under olika nätverksförhållanden kan en bättre förståelse fås för hur en webbapplikation presterar under mer krävande omständigheter. På så sätt kan tillförlitligheten till prestandan förstärkas. Detta kommer också att säkerställa att produkten levererar en konsekvent användarupplevelse oavsett vilken enhet den används på och i vilken arbetsmiljö.

Det andra kvalitetskravet fokuserade just på responsiviteten, vilket var avgörande för att garantera en konsekvent användarupplevelse över olika enheter och skärmstorlekar. Användningen av React och TailwindCSS möjliggjorde ett flexibelt och responsivt gränssnitt som uppfyllde dessa krav. Trots detta var förmågan att utnyttja verktygens fulla potential något begränsad av projektets tidsram. Med mer tid hade mer utforskning kunnat genomföras kring avancerade funktioner i TailwindCSS, som CSS Grid, för att skapa mer dynamiska och tvådimensionella layouter. Detta hade underlättat utvecklingsfasen för mer komplexa komponenter som tabeller. Hade CSS Grid använts hade tidsåtgången för att justera tabellens layout kunnat minskas då det ger mer kontroll över layoutstrukturen. Något annat som hade kunnat övervägas var att implementera Reacts kontext-API för att bättre anpassa gränssnittet till olika enhetstyper och skärmstorlekar. Detta är särskilt relevant för både datorer och surfplattor. Implementeringen av detta API hade kunnat förbättra användarupplevelsen ytterligare genom att göra gränssnittet mer responsivt under olika förhållanden. Även om kravet gällande responsivitet uppfylldes, kunde möjligheter till förbättringar identifieras som skulle kunna säkerställa optimal prestanda och användbarhet oavsett plattform.

Det tredje kvalitetskravet fokuserade på att maximera användarvänligheten genom att begränsa navigationsdjupet till högst åtta flikar. Detta mål var utformat med syfte att förenkla användarnas interaktioner med webbapplikationen, vilket är viktigt för att säkerställa att även de med begränsad teknisk erfarenhet kan använda den effektivt. Navigationsdjupet minskades till endast tre nivåer, vilket uppfyllde detta krav. Gruppen inledde med att se över ett initialt flödesschema över applikationen som kunden hade skapat. De viktiga funktionerna identifierades och omstrukturerades, vilket resulterade i en optimal och intuitiv trädstruktur. Denna effektiviserade användargränssnittet så att användarna enkelt kan nå viktig information utan att navigera genom onödiga flikar.

Det förenklade djupet har inte begränsat funktionaliteten. Diskussioner med kunden resulterade i en layout där samma mängd information nås med färre klick, vilket minskar risken

för frustration och förvirring, särskilt för användare med begränsad teknisk kunskap. Detta förbättrar både användbarheten och effektiviteten i användarnas interaktioner med systemet. Enkelheten är nyckeln till en förbättrad användarupplevelse. Framgången med detta kvalitetskrav belyser vikten av enkelhet i designen, något som är värdefullt att tänka på i framtida projekt.

Sammanfattningsvis visar denna diskussion om kvalitetskraven att även om de uppsatta målen uppnåddes finns det alltid rum för förbättring. Att identifiera möjligheter och kontinuerligt sträva efter att höja standarden på produktutvecklingen kommer att vara avgörande för fortsatt framgång.

6.2.2.2 Funktionella krav

Projektet fokuserade på att skapa ett användarvänligt och intuitivt gränssnitt, med syfte att uppfylla funktionella krav som bidrar till att förbättra användarinteraktionen och effektivisera arbetsprocesser på byggarbetsplatser. Centrala funktioner som inloggningsprocesser och ritningshantering implementerades, vilket var avgörande för att möjliggöra fortsatt utveckling och testning av systemet. Dessa funktioner stöds för närvarande av testdata, kallat ”dummy-data”, på grund av avsaknaden av en fullständig back-end, vilket var en känd begränsning redan från projektets start.

Fokus lades på att utveckla användarflödet för byggarbetare, vilket inkluderade funktioner som förenklade deras interaktion och tillgång till information. Bland funktionerna finns väderuppdateringar och möjligheter att ta genvägar till senast besökta ritningar, senast ändrade och favoritmarkerade ritningar, som i dagsläget inte leder till den aktuella ritningen då detta kräver back-end-integration. Dessa genvägar är avsedda att underlätta navigeringen direkt till den senaste ritningen man arbetade på eller hade skickat iväg ett ärende om, utan att behöva klicka igenom flera knappar. Användaren kan enkelt navigera genom sidomenyn till olika funktioner, där vissa klick leder till faktisk information medan andra indikerar att implementation saknas. Dessa funktioner som saknar fullständig implementation begränsades av projektets tidsram.

Även om framgång uppnåddes i funktioner som finns på byggarbetarnas flöde, begränsades förmågan att skapa en komplett administrativ sida på grund av tidsbegränsningar. Denna prioritering innebar att vissa funktioner som ärendehantering och stöd för ett önskat taggsystem inte har utvecklats. Det finns utrymme och ett uppenbart behov av ytterligare utveckling för att systemet ska vara heltäckande, ur både front-end och särskilt back-end-aspekt. Framåt är det viktigt att balansera utvecklingen för arbetare och administratörer så att alla användarbehov tillgodoses.

Genom att ha utvecklat front-end funktioner och komponenter som har skapat mervärde har projektet lyckats lägga en stabil grund för framtida utveckling. För att uppfylla alla funktionella krav och garantera en webbapplikation enligt kundens förväntningar krävs vidareutveckling som omfattar en fullständig back-end-lösning, vilket har nämnts tidigare. Framstegen som har gjorts och de insikter som tas med från detta kommer att vara avgörande i framtida projekt för att skapa en produkt som inte bara fungerar i teorin utan även i praktiska arbetsmiljöer.

6.2.2.3 Icke-funktionella krav

Implementeringen av de icke-funktionella kraven, som att designa estetiskt tilltalande menyer och knappar och inkluderingen av grundläggande flerspråkigt stöd, uppnådde syftet med att förbättra användarupplevelsen. Dessa krav lyckades bidra till ett mer intuitivt och lätthanterligt gränssnitt för användarna, vilket var målet. Några sådana förbättringar var tyd-

liga knappbeskrivningar, den minimerade navigationen mellan flikar och att snabbt kunna komma åt det som arbetades på senast.

Valet av det första kravet, att prioritera utseendet av menyerna och knapparna, var avgörande för att ge användarna ett system som var lätt att förstå sig på och att använda. Det var främst viktigt för att engagera användarna med mindre teknisk erfarenhet, vilket kan vara vanligt inom byggsektorn. Något som hade kunnat förbättra dessa användarinteraktioner hade varit om produkten var i ett skede där den kunde testas av användarna, särskilt i de arbetsmiljöer, förhållanden och på de enheter den är avsedd för.

Det andra kravet, om flerspråkigt stöd, är tänkt att bidra till lättare användning av systemet, då faktumet att en arbetare pratar ett annat språk inte ska behöva skapa utmaningar i arbetet. Det har skapats en grund för detta stöd genom front-end utvecklingen men den fulla potentialen för det här kravet kunde inte realiseras på grund av att back-end-integration saknas. En komplett sådan integration hade medfört ett dynamiskt språkbyte på sidan utefter användarens preferens.

Framöver är det viktigt att dessa krav inte bara ses som ett tillägg i projektet, utan som grundläggande komponenter som bör utvärderas och med det förbättras kontinuerligt. Det kan bidra till en förbättrad användarupplevelse och ett effektivt system. Till framtida projekt tas vikten av regelbunden feedback och användartester med för att anpassa systemet utefter användares behov och förväntningar.

6.2.3 Utvecklingsprocessen

I detta avsnitt diskuteras processen av projektets mjukvaruutveckling av webbapplikationen.

6.2.3.1 Kommunikation

Kommunikationen inom gruppen har spelat en avgörande roll för både framstegen och de utmaningar som gruppen stött på. Gruppen implementerade flera kommunikationsstrategier som var noga anpassade efter projektets behov, vilket inkluderade dagliga stand-up-möten och en noggrann dokumentation av arbetsinsatser via Google Sheets. Dessa metoder har generellt bidragit till ett högt mått av transparens och ansvar inom gruppen, vilket har varit avgörande för att upprätthålla en klar överblick över projektets framsteg och enskilda bidrag. Denna transparens i kommunikationen har underlättat för gruppmedlemmarna att ständigt vara uppdaterade om varandras arbete, vilket i sin tur har minskat risken för missförstånd och dubbelarbete. Den regelbundna och strukturerade koordinationen via stand-up-möten har också möjliggjort en effektiv problemlösning genom att snabbt identifiera och behandla frågor som kunde påverka projektets framdrift. Dessutom har den agila metodiken som kombinerar Scrum- och Kanban-principer tillåtit en stor flexibilitet i projektstyrningen. Denna flexibilitet har varit särskilt värdefull i ett projekt som detta, där föränderliga förhållanden och behov kräver en dynamisk arbetsprocess.

Trots alla fördelar har projektets starka beroende av digitala verktyg och plattformar för kommunikation och dokumentation även medfört vissa nackdelar. Ett överflöd av digital kommunikation kan ibland leda till tekniska störningar, vilket kan skapa hinder i informationsflödet och därmed försena viktiga beslutsprocesser. Det finns även en risk för informationsöverbelastning där viktig data kan bli förbisedd på grund av den stora mängden kommunikation som måste hanteras dagligen. Denna situation kan i värsta fall leda till att viktiga beslut fattas på en ofullständig informationsgrund, vilket potentiellt kan sänka projektets kvalitet.

Det kontinuerliga kravet på kommunikation och uppdateringar kan även leda till mötesutmattning bland gruppmedlemmarna. Om medlemmarna upplever att de frekventa mötena blir repetitiva och inte tillför värde, kan engagemanget och motivationen att bidra aktivt minska. Denna typ av utmattning är inte bara skadlig för individens välbefinnande, utan kan också ha en negativ inverkan på hela projektets dynamik och atmosfär.

För att bemöta dessa utmaningar krävs en balanserad kommunikationsstrategi som kan stödja projektets mål effektivt utan att bidra till onödig stress eller ineffektivitet. Det är viktigt att gruppmedlemmarna regelbundet reflekterar över och justerar sina kommunikationsmetoder. Att hitta rätt balans mellan tillräcklig och överflödig kommunikation är avgörande för att upprätthålla ett produktivt arbetsklimat och för att slutligen säkerställa ett framgångsrikt projektresultat.

6.2.3.2 Testning

Testprocessen i projektet genomgick förändringar under projektets gång, vilket påverkade hur olika typer av tester implementerades och utfördes. Gruppen stod inför utmaningar med den befintliga kodbasen och beslutade att utesluta enhets- och integrationstester för att spara tid och fokusera mer på utvecklingen. Detta beslut togs efter noggrant övervägande och diskussion inom gruppen. Detta visade sig vara ett bra val då det ledde till mer kodutveckling.

Även om enhets- och integrationstester exkluderades genomfördes systemtester enligt planen med hjälp av Cypress för att säkerställa att kraven för projektet uppfylldes. Denna strategi gav gruppen möjlighet att fokusera på att validera den slutliga funktionaliteten och användarupplevelsen.

Automatiserade tester, särskilt med verktygen Lighthouse och Cypress, spelade en central roll i testprocessen. Gruppen satte upp automatiserade tester i början av projektet, med hjälp av GitHub Actions, för att säkerställa att nya kodförändringar inte introducerade oväntade problem eller försämrade prestandan. Utmaningar uppstod dock när begränsningar för tester i GitHub Actions ledde till att gruppen behövde hitta alternativa lösningar för att fortsätta med automatiserade tester. Att använda en egen server för att köra tester visade sig vara en kreativ och bra lösning på detta problem, då det möjliggjorde fortsatt automatiserad testning av varje pull-förfrågan och varje push till grenarna `dev` och `main`.

Slutligen, trots de utmaningar och förändringar som testprocessen genomgick, lyckades gruppen ändå säkerställa att kvaliteten på den slutliga produkten uppfyllde kraven och förväntningarna. Genom att prioritera systemtester och implementera automatiserade tester kunde gruppen effektivt validera funktionalitet och säkerställa en tillfredsställande användarupplevelse.

6.2.3.3 Versionshantering

Git är ett effektivt verktyg som gruppen huvudsakligen använde i versionen kallad GitHub. Valet föll på GitHub eftersom den existerande koden från kunden fanns på GitHub. Gruppen hade tidigare nästan uteslutande använt en annan verision av Git i sina studier kallad GitLab. Erfarenheten av GitHub var positiv och gruppen hade möjlighet att lära sig om de fördelar och nackdelar som GitHub erbjuder.

Det genomfördes automatiska tester av applikationen, vilket överlag gick bra. Dock inträffade en incident där det tog slut på tid för att köra testerna i den molnbaserade lösningen, på grund av en felkonfiguration i GitHub. Detta löstes genom att köra testerna på egen server.

Det stöttes på vissa problem med pull-förfrågningar, där konflikter uppstod mellan den befintliga kodbasen och den nya koden som skulle integreras. Det kunde dock enkelt lösas med manuell granskning på GitHub.

Gruppen använde även den inbyggda Kanban-tavlan som finns tillgänglig på GitHub, vilket gav blandade resultat. Halvvägs in i projektet slutade gruppen använda den för att visualisera vad som arbetades med och övergick istället till att använda dokument för att representera pågående uppgifter då detta uppfattades enklare.

6.2.3.4 Dokumenthantering

När det kommer till dokumenthantering har gruppens erfarenhet med Overleaf varit mycket positiv. Jämfört med andra dokumenthanteringsprogram krävde Overleaf en större inlärningsinsats för att fullt ut förstå dess funktioner. Vissa utmaningar stöttes på när det gällde att skapa figurer och tabeller, särskilt med tanke på de omfattande formateringsreglerna i LaTeX. Efter en utbildnings process i Overleafs funktioner och krav, kunde gruppen dock ta an dessa utmaningar. Resultatet är att samtliga dokument presenteras med en hög estetisk standard.

6.2.4 Parprogrammering

Gruppen anser att implementeringen av parprogrammering har haft en betydande positiv inverkan på både kodkvaliteten och samarbetsdynamiken inom gruppen. Denna metod bidrog inte bara till färre kodfel utan också till en mer inkluderande och stödjande arbetsmiljö. Det gemensamma arbetssättet underlättade kunskapsöverföring och erfarenhetsdelning, vilket var särskilt värdefullt i ett lärande sammanhang som detta kandidatarbete.

Parprogrammering hade kunnat vidareutvecklats eller anpassats för att ytterligare öka dess effektivitet. Till exempel skulle strukturerade granskningstillfällen eller ökad rotation av vilka som är i par kunnat undersökas som metoder för att ytterligare bredda kompetensen inom gruppen. På detta sätt har gruppen sett parprogrammering som inte bara en teknik för kodutveckling utan som en grundläggande princip för att bygga en stark och kompetent grupp vilket var viktigt för att kunna slutföra projektet.

6.2.5 Kundkontakt

Under projektets gång fick kommunikationsmetoden anpassas utifrån det som passade gruppen och kunden. Den huvudsakliga kommunikationskanalen var e-post, vilket vanligtvis inte hade varit ett förstahandsval då svarstider kan vara långa. Det fungerade dock bra då både kunden och analysansvarig var snabba med att svara. Den utmaning gruppen mötte med denna kommunikationskanal var i början av projektet när kundens e-post inte fungerade, vilket medförde en längre svarstid. Detta problem löstes genom att analysansvarig alltid skickade iväg e-postmeddelande till två av kundens e-postadresser, på så sätt kunde det säkerställas att ingen e-post missades på grund av tekniska skäl.

Kunden hade också ordnat för användning av Microsoft Teams för kommunikation, men det användes främst för att dela filer och dokument snarare än att kommunicera. Det var en överenskommelse som gjordes tillsammans då den initiala kommunikationskanalen, e-post, fungerade alldeles utmärkt för den direkta kommunikationen. Valet att inte använda Microsoft Teams för direkt kommunikation var främst baserat på kundens preferens, som upplevde att det fanns andra kanaler som bättre passade syftet.

Mot projektets slut skapades även en WhatsApp-grupp utifrån kundens önskan, dels för att ta lite snabbare frågor men även för att alla gruppmedlemmar kunde både skriva och ta del

av andras frågor och svar. Kunden var väldigt tillgänglig och flexibel vilket bidrog till en bra kommunikation. Möjligheten att inte bara skriva, utan även ringa kunden, när som helst bidrog till att effektivisera arbetet då gruppen vid akuta frågor kunde ringa och få svar direkt och på så vis ta snabba beslut.

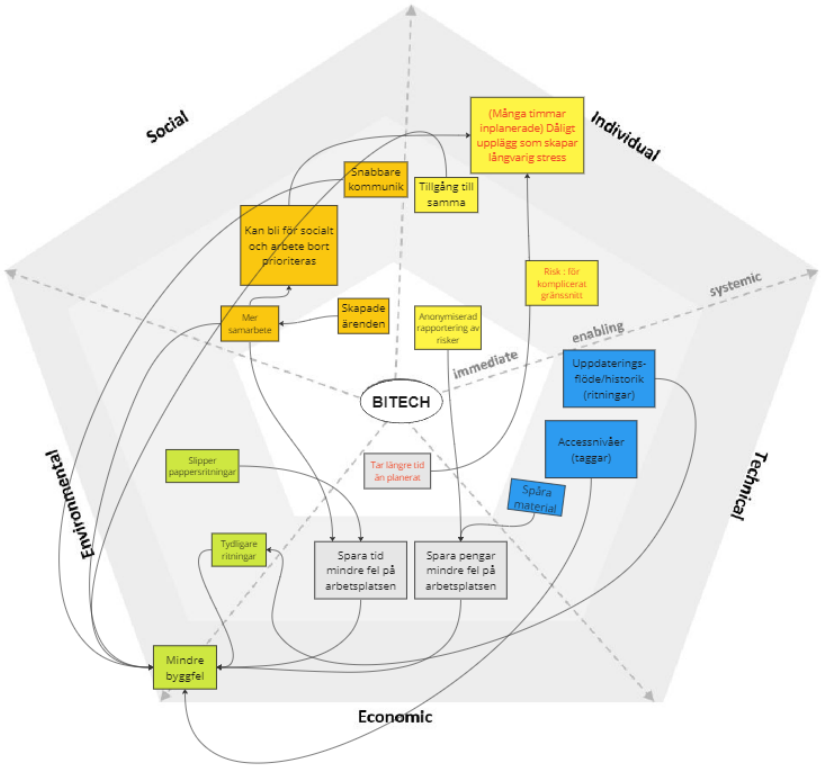
Det gruppen tar med sig från detta är vikten av flexibla och anpassningsbara kommunikationsmetoder då olika kanaler kan fungera olika bra för olika kunder. I detta fall fungerade e-post, men något att ha i åtanke är att inte alla kunder kommer att vara lättillgängliga och flexibla som denna kund. Därför kan det vara bra att i framtida projekt överväga ett mer integrerat tillvägagångssätt redan från början så att den direkta kommunikationen sker via en kanal som främjar snabbhet och flexibilitet, även om kunden själv inte hade varit lika snabb.

6.2.6 Systemet

Genom att digitalisera processen för ritningshantering kan samarbetet förbättras mellan olika aktörer inom byggprojekt, från designers till entreprenörer, vilket kan leda till minskade kostnader, förbättrad tidsplanering av projekt och en minskning av fel som orsakas på grund av felaktiga eller föråldrade ritningar. Därmed tror gruppen starkt på att denna applikation bör utvecklas vidare av andra studenter just för att systemet är lättbegripligt samtidigt som det har en positiv effekt på samhället. Systemet har en stor variation av problem som behöver lösas och har möjlighet att skalas upp i internationell skala vilket kan vara av intresse för framtida utvecklare. Utformandet av system gjordes med hjälp av kundmöten där krav skapades enligt kundens idéer, se avsnitt 5.3 för mer detaljer.

6.3 Arbete i ett vidare sammanhang

I detta avsnitt diskuteras produktens påverkan på samhället där applikationen antas vara helt fullbordad och komplett i alla aspekter. För att få in flera perspektiv användes *susAF*-metoden för att granska följande aspekter: hållbarhet, sociala, individuella, tekniska och ekonomiska. Samtliga aspekter finns sammanfattade i figur 6.1 där de är visualiserade i ett *susAD*-diagram.



Figur 6.1: SusAD-diagram för projektet

6.3.1 Hållbarhetsaspekter

Miljömässigt har denna applikation positiva och strukturella konsekvenser då den minskar användadet av pappersritningar. Det kan dock finnas en viss mängd pappersritningar som fortsätts att användas exempelvis om alla inte får tillgång till en surfplatta. Detta hade kunnat lösas genom att till exempel göra applikationen anpassad för smarta mobiltelefoner vilket de allra flesta arbetarna har tillgång till. Visserligen kan det observeras att användningen av ett ökande antal mobila enheter, till exempel surfplattor, kan ha en negativ miljöpåverkan. Detta beror på att tillverkningen av dessa enheter kräver betydande resurser. Trots denna insikt har gruppen beslutat att inte fördjupa sig ytterligare i denna specifika fråga.

6.3.2 Tekniska aspekter

De tekniska aspekterna och konsekvenserna för applikationen är att ta hänsyn till. Exempelvis kräver hanteringen av digitala ritningar tillgång till pålitlig teknik. Det gäller nämligen internetanslutning och säkra datalagringslösningar. Övergången till digitala ritningar innebär att känslig information lagras elektroniskt. Då krävs det att robusta säkerhetsåtgärder implementeras för att skydda ritningarna från obehörig åtkomst, dataskador eller andra förluster.

6.3.3 Etiska aspekter

De etiska aspekterna av webbapplikationens användning på marknaden är främst två centrala frågor. Det gäller den potentiella arbetslösheten och datasäkerheten. Webbapplikationen, avsedd att digitalisera och effektivisera arbetsprocesser inom byggprojekt, kan innebära en risk för att vissa traditionella roller inom byggsektorn blir mindre efterfrågade eller försvinner. Det är därför viktigt att överväga hur tekniken implementeras. Däremot, gällande säkerhetsaspekterna, är det avgörande att applikationen upprätthåller höga säkerhetsstandarder för att skydda känslig information som hanteras inom byggprojekten. Detta inkluderar att säkerställa att datakryptering och åtkomstkontroller är på plats för att förhindra obehörig åtkomst och dataintrång. Genom att behandla dessa etiska frågor, kan webbapplikationen inte bara förbättra effektiviteten i byggprojekten utan också säkerställa att övergången till digitala verktyg sker på ett socialt ansvarsfullt och säkert sätt.

6.3.4 Socioekonomiska konsekvenser

Om systemet vidareutvecklas och fullbordas kan det ha betydande socioekonomiska effekter. Genom att motverka vanliga fel i byggprojekt kan systemet spara stora belopp, vilket förbättrar kostnadseffektiviteten och lockar fler investerare till byggbranschen. Ur kundperspektiv innebär detta kortare byggtider och minskade kostnader, vilket ökar tillgången till prisvärda bostäder och infrastruktur. Systemet främjar en effektivare arbetsmiljö, vilket inte bara förbättrar villkoren för arbetare och platschefer utan även gör byggbranschen mer attraktiv för ny arbetskraft.

Samtidigt måste vi beakta miljöpåverkan av ökade byggnationer. Det är viktigt att säkerställa att material som används har låg miljöpåverkan och att byggandet är hållbart. Systemet kan bidra till detta genom att optimera resursanvändningen och minska slöseri. Om vi inte bygger utifrån verkliga behov riskerar vi att överutnyttja resurser och skapa onödiga strukturer. Därför är det avgörande att systemet integrerar hållbarhetskriterier och behovsanalys för att balansera tillväxt och miljöhänsyn på ett ansvarsfullt sätt.

6.3.5 Individuella Aspekter

Applikationen påverkar arbetarnas arbetsmiljö och effektivitet på flera sätt. Den förbättrar tillgången till information genom att ge alla medarbetare samma förutsättningar att komma åt relevant data, vilket främjar transparens och samarbete. Dock finns det risker förknippade med komplexa gränssnitt som kan leda till stress och långvariga arbetsbelastningar om användarvänligheten inte beaktas. Dessutom möjliggör applikationen anonym rapportering, vilket stärker säkerheten och arbetsmiljön genom att uppmuntra rapportering av problem utan rädsla.

6.4 Källkritik

Källorna som hänvisas till i denna rapport kan variera i kvalitet, men samtliga har undergått en noggrann granskning för att säkerställa deras tillförlitlighet och relevans för ämnet. Denna granskning har utförts med hjälp av flera kriterier, författarens auktoritet och expertis i fokus men även källans trovärdighet och objektivitet, samt dess relevans för rapportens syfte och mål.

För att erhålla information om olika verktyg har generellt respektive verktygs hemsida använts som källa. Dessa källor förväntas ge den mest aktuella och pålitliga informationen om sina egna produkter och tjänster. Genom att använda deras egna hemsidor som källor kan tillgång till den senaste uppdaterade informationen säkerställas. Det är dock viktigt att notera att information från verktygens egna hemsidor kan vara vinklad, eftersom dessa källor

kan ha en tendens att framhäva fördelarna med sina produkter medan nackdelar och begränsningar inte alltid framgår tydligt. Därför har vi, där det varit möjligt, kompletterat med information från oberoende källor för att få en mer balanserad bild.

Publicerade artiklar från konferenser och vetenskapliga tidsskrifter har använts för att stärka andra delar av rapporten. Dessa artiklar genomgår vanligtvis en noggrann granskningsprocess innan de publiceras, vilket ger en hög grad av tillförlitlighet inom det aktuella området. Dessutom har den konferens eller tidsskrift där artikeln är publicerad granskats av gruppen för att säkerställa dess rykte och akademiska integritet, innan källan bedömts som tillförlitlig och relevant för rapporten.

7 Slutsatser

Detta avsnitt presenterar slutsatserna som kunde dras av frågeställningarna som finns i kapitel 1.3.

7.1 Hur kan hemsidan implementeras så att man skapar värde för kunden?

Värde har skapats och främst genom att anpassa hemsidan för surfplattor så användarna kan dra nytta av systemets bärbarhet. Detta gör det möjligt för användare på byggarbetsplatser att enkelt få tillgång till och navigera i digitala ritningar och dokument utan att kompromissa med användbarheten. Detta stödjer en snabb och effektiv informationsdelning på fältet, vilket blir avgörande för projektet. Dessutom finns redan en etablerad standard i byggarbetsplatser där surfplattor används, så att implementera detta ger mycket mervärde för kunden.

För mobila enheter, med tanke på deras allmänna tillgänglighet och användning, är det viktigt att hemsidan är lättillgänglig och funktionell även på mindre skärmar. Detta ser till att all personal, oavsett deras specifika roll eller plats, har tillgång till nödvändig projektinformation när som helst. Vilket ger värde i form av förbättrad kommunikation och effektivitet över hela linjen.

Slutligen: Att anpassa hemsidan för användning på datorer är avgörande för administrativ och ledningsmässig personal. Datorer ger en mer robust plattform för att hantera omfattande projektdata, köra detaljerade analyser och underhålla omfattande databaser. Att optimera hemsidan för datoranvändning tillåter platschefer och administratörer att utföra mer komplexa uppgifter, till exempel ansvars- och arbetsfördelning mer effektivt.

7.2 Vilka erfarenheter kan dokumenteras från programvaruprojektet som kan vara intressanta för framtida projekt?

Inledningsvis har projektet understrukit betydelsen av effektiv kommunikation och sammarbetsverktyg. Användningen av Discord som huvudsaklig intern kommunikationskanal har exempelvis möjliggjort omedelbar dialog mellan gruppmedlemmarna, vilket har varit särskilt fördelaktigt under perioder av distansarbete. Denna plattform har inte bara underlättat textbaserad kommunikation utan också erbjudit möjligheter för röst- och videosamtal.

Regelbundenheten i gruppens möten, till exempel stand-up möten, arbetspass och gruppmöten, har spelat en stor roll för projektets framgång. Detta genom att stödja en stabil struktur för uppgiftsdelegering och synkronisering av arbetet. Dessa möten har sett till att alla gruppmedlemmar varit kontinuerligt uppdaterade och engagerade i projektets mål, vilket i sin tur har bidragit till en högre effektivitetsnivå genom hela projektet.

Dessutom har aktivt lyssnande och engagemang i kundinteraktioner varit avgörande för att exakt tolka och tillgodose kundens krav. Regelbundna och strukturerade kundmöten har stärkt både förståelsen för projektets mål och relationen med kunden, vilket är en praxis som starkt rekommenderas för framtida projekt.

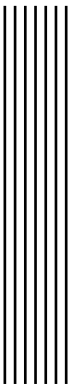
7.3 Vilket stöd kan man få genom att skapa och följa upp en systemanatomi?

I projektet har användningen av systemanatomi visat sig vara avgörande för att uppnå en effektiv och koordinerad arbetsprocess. Genom denna praktik kunde projektgruppen upprätthålla en klar struktur genom alla faser av projektet, vilket direkt stödde både individuellt och kollektivt arbete. Effekterna av detta stöd var tydliga i vår förmåga att snabbt identifiera och åtgärda problemområden, vilket ledde till färre förseningar och en mer tidseffektiv projektgenomförande. Dessutom underlättade systemanatomin kommunikationen inom gruppen och med externa intressenter, vilket var kritiskt för att hålla alla informerade och engagerade i projektets framsteg.

Genom att skapa och följa upp en systemanatomi kunde gruppmedlemmarna lättare få en översiktlig struktur för systemet vilket hjälpte gruppen att designa systemet med varandras arbete i åtanke. Anatomin hjälpte gruppen att designa systemet med en uppdelning av medlemmarnas arbete i åtanke. Det underlättade tilldelningen av uppgifter och visade på interaktionen mellan olika komponenter vilket stödjer medlemmarna då alla kan hålla sig till en mall. När systemet har nått flera viktiga milstolpar i sitt utvecklingsprojekt kan man även få stöd av anatomin för att se möjligheter till vidareutveckling.

7.4 Vilken inverkan kan ett digitalt ritningshanteringsprogram ha på byggindustrin?

Projektet presenterar nyskapande lösningar för att förbättra kommunikation, hantering av material och säkerhet på byggarbetsplatser. Genom direkt kommunikation mellan arbetare och chefer, GPS-spårning av material och digital hantering av dokument främjas effektivitet och transparens. Säkerheten förbättras genom begränsad åtkomst till känslig information och tillgång till digitala säkerhetsinstruktioner. Sammantaget kommer dessa åtgärder att leda till en mer effektiv och säker byggindustri med minskade kostnader och smidigare arbetsflöden.



Litteratur

[1] David Grossman. ”Byggbolagen måste bli mer produktiva”. I: *Fastighetstidningen* (juni 2023). https://fastighetstidningen.se/byggbolagen-maste-bli-mer-produktiva/.

[2] Kristian Sandahl. *TDDD96, Kandidatprojekt i programvaruutveckling*. https://www.ida.liu.se/~TDDD96/inbjudan/index.sv.shtml. 2023. (Hämtad 2024-03-30).

[3] ”IEEE Recommended Practice for Software Requirements Specifications”. I: *IEEE Std 830-1998* (1998), s. 1–40. DOI: 10.1109/IEEESTD.1998.88286.

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

[5] ”IEEE Standard for Software and System Test Documentation”. I: *IEEE Std 829-2008* (2008), s. 1–150. DOI: 10.1109/IEEESTD.2008.4578383.

[6] Chuck Musciano och Bill Kennedy. *HTML & xhtml: The definitive guide: The definitive guide*. O’Reilly Media, Inc., 2002, s. 15–35.

[7] Hakon Wium Lie och Bert Bos. *Cascading style sheets: Designing for the web*. Addison-Wesley Professional, 2005, s. 33–52.

[8] David Flanagan. *JavaScript: The definitive guide: Activate your web pages*. O’Reilly Media, Inc., 2011, s. 1–20.

[9] *What is Vue?* https://vuejs.org/guide/introduction#what-is-vue. 2024. (Hämtad 2024-05-02).

[10] *React*. https://react.dev/. 2024. (Hämtad 2024-03-05).

[11] Sanchit Aggarwal m. fl. ”Modern web-development using reactjs”. I: *International Journal of Recent Research Aspects* 5.1 (2018), s. 133–137.

[12] Ricky Jonathan och Suprihadi. ”Development of Front-End Web Applications Utilizing Single Page Application Framework and React.js Library”. I: *International Journal Software Engineering and Computer Science (IJSECS)* 3.3 (2023), s. 529–536. ISSN: 2776-4869, 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v3i3.1943.

[13] TailwindCSS. *Rapidly build modern websites without ever leaving your HTML.* https://tailwindcss.com/. 2024. (Hämtad 2024-03-05).

' [14] *React testing library*. https://testing-library.com/docs/react-testing-library/intro/. 2024. (Hämtad 2024-03-05).

[15] *Jest*. https://jestjs.io/. 2024. (Hämtad 2024-01-31).

[16] *Why Cypress?* https://docs.cypress.io/guides/overview/why-cypress. 2024. (Hämtad 2024-05-03).

[17] *Lighthouse CI*. https://github.com/GoogleChrome/lighthouse-ci.

[18] Eduard Pangestu Wonohardjo, Rizky Febriyanto Sunaryo och Yusuf Sudiyono. "A systematic review of scrum in software development". I: *JOIV: International Journal on Informatics Visualization* 3.2 (2019), s. 108–112.

[19] Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy och Carol Zander. "Pair programming in education: A literature review". I: *Computer Science Education* 21.2 (2011), s. 135–173.

[20] Nevenka Kirovska och Saso Koceski. "Usage of Kanban methodology at software development teams". I: *Journal of applied economics and business* 3.3 (2015), s. 25–34.

[21] Stefanie Betz, Leticia Duboc, Birgit Penzenstadler, Jari Porras, Ruzanna Chitchyan, Norbert Seyff, Colin C. Venters och Ian Brooks. *SusAF Workbook 6.0*. Version 3. Dec. 2022. DOI: 10.5281/zenodo.7342575.

[22] *What is Next.js?* https://nextjs.org/docs#what-is-nextjs. 2024. (Hämtad 2024-05-03).

[23] *What is Figma?* https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma. 2024. (Hämtad 2024-05-03).

[24] Jon Loeliger och Matthew McCullough. *Version Control with Git: Powerful tools and techniques for collaborative software development*. O'Reilly Media, Inc., 2012, s. 19–46.