CHALMERS

LatPlan

Utforskande av element i en kalenderapplikation för ökad användbarhet

Examensarbete inom högskoleingenjörsprogrammet datateknik

Emil Johansson

Sebastian Bergdahl

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA

Göteborg, Sverige 2024
www.chalmers.se

Examensarbete 2024

LatPlan

Utforskande av element i en kalenderapplikation för ökad användbarhet

Emil Johansson
Sebastian Bergdahl

CHALMERS

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2023

LatPlan
Utforskande av element i en kalenderapplikation för ökad användbarhet

Handledare: Patrik Jansson, D&IT
Examinator: Jonas Duregård, D&IT

Examensarbete 2024
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Institutionen för data- och informationsteknik
Göteborg 2023

LatPlan
Utforskande av element i en kalenderapplikation för ökad användbarhet
EMIL JOHANSSON
SEBASTIAN BERGDAHL
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola

Sammanfattning

Digitala kalenderapplikationer används idag av en stor del av befolkningen men har brister som förhindrar vissa grupper från att utnyttja dess fördelar. Projektet syftar till att konstruera en grundläggande kalenderapplikation, LatPlan, i syfte att testa funktioner som ökar mobila kalenderapplikationers användbarhet.

LatPlan konstrueras med den funktionalitet som krävs för att schemalägga en kalender. Därefter implementeras flera olika användbarhetsökande funktioner inom tre kategorier. För att utvärdera användbarheten av de implementerade funktionerna utförs en undersökning med hjälp av testare. Undersökningen är baserad på Nielsens heuristiska principer. Principerna är utvecklade utifrån behovet av att jämföra användbarheten mellan olika lösningar

Med hjälp av resultaten från undersökningen utvärderas lösningar gentemot andra inom samma kategori. Därefter presenteras de lösningar som anses bäst inom respektive kategori, samt i vilken utsträckning de anses öka användbarheten vid användandet av digitala kalendrar.

Abstract

Digital calendar applications are widely used today, but they have limitations that prevent certain groups from fully benefiting from them. The project aims to develop a basic calendar application, LatPlan, in order to test features that enhance the usability of mobile calendar applications.

LatPlan is designed with the necessary functionality for scheduling a calendar. Subsequently, several usability-enhancing features are implemented across three categories. To evaluate the usability of the implemented features, a survey is conducted with the help of testers. The survey is based on Nielsen's heuristic principles, which are developed to compare the usability of different implementations.

Using the results from the survey, the implementations are evaluated against each other within the same category. The best implementations within each category are then presented, along with the extent to which they are considered to improve the usability of digital calendars.

Nyckelord: Kalender, Användbarhet, Användarupplevelse

Innehåll

1 Introduktion

Kalendrar är en vanlig lösning på problemet som är att hantera och planera den tid som en användare har tillgänglig. Att kunna planera saker med hjälp av en kalender bidrar till allt ifrån en effektivare arbetsdag till en mer effektiv familjeplanering [1]. Studier visar att upp till 70 procent använder en digital kalender och omkring hälften gör så med hjälp av sin mobil [2].

Användandet av en digital kalender har många fördelar såsom att den kan nås så länge man kan koppla upp sig till ett nätverk och att man inte behöver bära runt på en fysisk kalender. Men det finns negativa aspekter med att använda en digital kalender gentemot en fysisk. Planer som har gjorts på papper har en större sannolikhet att genomföras än de som har gjorts digitalt [1]. En annan stor fördel med att göra planer icke-digitalt är att anteckningar kan göras så kortfattade eller utförliga som användaren önskar medan en digital kalender har krav på vilken information som måste ges för att informationen ska sparas. Dessa krav skapar en försämrad användbarhet gentemot förmågan att flexibelt kunna spara informationen för mer personlig planering.

För att mitigera bristerna hos en digital kalender samt öka användbarheten kommer olika användbarhetsökande metoder implementeras och utvärderas. Detta för att kunna underlätta för en snabb planering när tiden inte finns för att skriva ner alla detaljer eller när all information inte finns. Hjälpen för att snabbt planera saker underlättar framför allt för vissa grupper som har ett stort behov av att planera men kan ha svårt för att göra det, exempelvis individer som är diagnostiserade med ADHD [3].

Ökandet av användbarhet kan utföras inom många områden av en applikation. För att arbetet ska främja inklusivitet och social hållbarhet så kommer rapportens fokus att ligga hos användaren som inte vanligtvis använder en kalender på grund av bristande användbarhet, till skillnad från att underlätta övergången för en grundläggande användare till en avancerad.

1.1 Syfte

För att underlätta planeringen och kartläggningen av användarens tid ska funktioner för ökandet av en kalenders användbarhet undersökas. Applikationen LatPlan ska konstrueras som en grund för testandet av funktionerna.

1.2 Mål

Målet med projektet är att undersöka användbarhetsökande metoder för att underlätta användandet av digitala kalenderapplikationer för grupper som idag inte använder digitala kalendrar genom att besvara två frågeställningar:

- Kan alternativa funktioner implementeras för att öka användbarheten av digital kalenderapplikationer?
- Vad är bra implementationer för att utföra befintliga funktioner som ökar användbarheten av digital kalenderapplikationer?

För att svara på frågeställningarna behövs en grundläggande kalenderapplikation som konstrueras med den funktionalitet som tillåter skapandet, sparandet och visandet av ett schema.

1.3 Avgränsningar

Huvudmålet med examensarbeten är att bidra till det akademiska kollektivet, inte att utveckla en produkt, vilket medför att en del avgränsningar tillkommer för att en fungerande applikation ska hinna konstrueras.

LatPlan ska endast utvecklas med stöd för iOS och Android för att begränsa den mängd ramverk som behöver hanteras. Det ramverk som ska användas är Flutter med programspråket dart för att tillåta utveckling mot två olika plattformar samtidigt.

Appens språk kommer att begränsas till enbart svenska. Appen kommer enbart stödja Sverige som geografiskt område. Tidzoner samt sommar/vintertid utanför CET/CEST kommer inte att hanteras. Datum och tidsformat begränsas till svensk standard.

Regler och lagar angående persondata kommer att undersökas men eftersom LatPlan inte är tänkt för en marknad så kommer inte stöd för hantering av exempelvis GDPR att implementeras.

Algoritmen som används i LatPlan kommer inte använda sig av någon form av generativ AI utan endast implementera en preferensbaserad inlärnings-algoritm.

För att kunna undersöka implementationen av användbarhetsökande funktioner, med den begränsade utvecklingstid som projektet har, så kommer inte alla lösningar att implementeras. Funktionalitet som fokuserar på enkel och snabb planering samt inmatning kommer huvudsakligen att implementeras. Övrig funktionalitet kommer att diskuteras och de metoder som anses effektiva kommer att rekommenderas för vidareutveckling.

2 Teknisk bakgrund

Rapporten kommer att nämna ett flertal begrepp samt mjukvaror vilka förklaras i detta kapitel. De har delats upp i två avsnitt där det första förklarar de begrepp som används i rapporten och det andra förklarar de mjukvaror som används under projektets gång.

2.1 Begrepp

Programmering görs huvudsakligen på engelska så många av de termer som har uppstått kring området översätts inte direkt till svenska. Därför har vi en samling av de termer som upprepas frekvent i rapporten med en förtydligande definition för att förvirring ska undvikas. Även om terminologin ibland följer en svensk ordboks definition så kan de förklaras ytterligare då läsaren inte förväntas ha en ordbok nära till hand och inte ska behöva jämföra olika definitioner.

2.1.1 Användbarhet

Under slutet av 70-talet började man studera hur människor interagerar med datasystem, vilket gavs namnet mjukvarorpsykologi (software psychology)[4]. Disciplinen byggde på experimentell psykologi i kombination med vetenskaplig tradition, där empirisk data samlades in med hjälp av kontrollerade experiment för att skapa teorier. Teorierna syftar på att förutsäga resultat för nya situationer utan att behöva genomföra nya studier[5].

Efterhand som dessa teorier testades så insåg man att de inte påverkade praktisk systemutveckling i den utsträckning som önskats. I praktiken så var kraven på kostnadseffektivitet och projektstyrning mycket högre än i teorin vilket formade om disciplinen till vad som kallas användbarhetskonstruktion (usability engineering), som kombinerar experimentell psykologi med samlandet av information. Användbarhetskonstruktion utgår ifrån ett ingenjörsmässigt tankesätt där man kvantitativt specificerar egenskaper man vill ha och sedan efter konstruktion verifierar att de har implementerats[5].

Nästan precis tio år efter disciplinens uppkomst introducerades begreppet användbarhet med sina första definitioner[5]. Med åren så har definitionen skiftat och idag definieras det som ett mått för att bekräfta att användare samspelar med designen på det sätt som förväntas samt att användarupplevelsen är praktisk, brukbar och önskvärd [6]. Användbarhet kan därmed användas som ett mätbart värde genom att utföra undersökningar som bygger på de tidigare psykologiska principerna.

För att mäta användbarhet så utförs användbarhetstester som är mest effektiva på att testa namngivning, struktur, upptäckbarhet och effektivitet [6]. Beroende i vilket skede av utvecklingsprocessen man befinner sig i så är olika tester mer eller mindre lämpliga. Summeringsutvärderingar används ofta i slutet av utvecklingsprocesser för att jämföra lösningar. Ett exempel på en summeringsutvärdering är Nielsens heuristiska utvärdering [6].

2.1.2 Nielsens heuristiska utvärdering

I takt med att gränssnittsdesign tog stora framsteg inom förbättrandet av användbarhet på 90-talet så ansåg Jakob Nielsen, i samarbete med Rolf Molich, att det fanns ett behov för en metod att analysera och hitta problem med gränssnittslösningar. För att bedöma ett gränssnitts användbarhet utvecklades en metod där ett fåtal testare kan på ett effektivt sätt identifiera en stor del av problematiken med ett gränssnitt [7]. Metoden bygger på att användare först tränas på applikationen och sedan får utföra uppgifter som de utvärderar utifrån olika kategorier. Dessa kategorier baseras på tio heuristiska punkter kring gränssnittsdesign, där de punkter som anses relevanta används för att formatera de frågor som ställs till användaren [8].

Principerna från de tio heuristiska punkterna har under åren förblivit samma efter en revision 1994[8], efter att en studie med 249 användbarhetsproblem utfördes [9], men språket för att beskriva dem har moderniserats. Punkterna som de heter idag är [7][8][10]:

1. Synlighet av systemtillstånd (Visibility of system status): Håll användaren informerad kring systemets tillstånd och ge återkoppling till användaren.
2. Systemet och verkligheten ska överensstämma (Match Between the System and the Real World): Funktioner ska fungera som de hade förväntats fungera om de hade varit verkliga och språk ska vara grundat i ord och fraser som en användare kan förväntas att kunna.
3. Användarkontroll och frihet (User Control and Freedom): Användaren ska enkelt kunna göra om och avbryta handlingar. Dessa funktioner ska vara tydligt markerade.
4. Följdriktighet och standarder (Consistency and Standards): Ord och namn ska standardiseras, därmed ska olika ord inte betyda samma sak och ett ord ska inte betyda flera saker.
5. Förhindrande av fel (Error Prevention): Den bästa felhanteringen är att fel inte sker, vilket innebär att användaren förhindras att skapa problematik för sig själv.
6. Igenkänning över ihågkommande (Recognition Rather than Recall): Användaren ska inte behöva komma ihåg information från tidigare steg, utan information som behövs ska fortsätta vara synlig.
7. Flexibilitet och effektivitet av användande (Flexibility and Efficiency of Use): Design ska möjliggöra för avancerade användare att påskynda långsamma och frekventa uppgifter genom ytterligare gömd funktionalitet som inte hindrar en oerfaren användare.
8. Utseende och minimalistisk design (Aesthetic and Minimalist Design): Information ska vara relevant och gränssnitt ska endast innehålla nödvändig information och funktionalitet då överflödiga alternativ minskar den relativa synligheten av viktig information.
9. Hjälp användare känna igen, diagnosticera och återhämta sig från fel (Help Users Recognize, Diagnose, and Recover from Errors): Felmeddelanden ska vara tydliga och förståeliga för användare samt tillåta en användare att återhämta sig från ett problem.
10. Hjälp och dokumentation (Help and Documentation): Ett system ska helst inte kräva någon manual, men en manual kan vara nödvändig för att hjälpa en användare för att klara sina uppgifter vid undersökningen. Manualen ska vara lätt att hitta information i och om den används så ska det noteras när och för vilka steg.

I en studie för att jämföra Nielsens heuristiska utvärdering med andra metoder konstaterade Nielsen att efter fem testare så hittade ytterligare testare resultatet avsevärt mindre ytterligare problem. De första fem testarna hittade tillsammans mellan femtio och nittio procent av alla problem, beroende på vilken applikation som testats. När en undersökning gick från fem testare till tio testare så hittades endast genomsnittligen 5 procentenheter ytterligare problem och skillnaden mellan tio och femton testare var så liten som 3 procentenheter [9].

2.1.2 Event

I projektet specificeras event som en händelse i schemat. Varje event har en titel samt start- och slut-tid med minutprecision. Ett event kan innehålla mer information och ha flera ytterligare attribut som fast och lös planering.

2.1.3 Fast och löst planerat

Event i projektet har egenskapen att vara flaggade som fast eller löst planerade. Fast planerade event är ett event där start- och sluttid specificerades av användaren vid skapandet av eventet. Löst planerade event planeras istället av en algoritm och är flaggade som löst planerade.

2.1.4 Schema

En samling av event(s) refereras till som ett schema. Storleken av schemat beror på hur många dagar som användaren vill visa på skärmen där de tre vanligaste alternativen är 1 dag (dagsschema), 7 dagar (veckoschema) eller 28 till 31 dagar (månadsschema). Ett schema kan vara tomt då avsaknandet av sparad information fortfarande är intressant för en användare

2.1.5 AI och inlärnings-algoritm

AI som begrepp används ofta istället för maskininlärning när man vill beskriva en algoritm som tränas på ett dataset för att sedan kunna ta beslut med hjälp av ny data. Detta skiljer sig från vad som hänvisas till som en inlärnings-algoritm i rapporten som istället endast lär sig av vad användaren gör.

2.1.6 GDPR

För att skydda EU-medborgares data som samlas in av företag så införde EU en föreskrift som kallas GDPR. Den har en bred omfattning och kan antas påverka alla delar av en organisation som samlar in persondata. Personlig data definieras som all data som kan knytas till en person med hjälp av en enda identifierare, exempelvis personnummer, adress och e-post. Data som kan knytas till en individ kan anonymiseras innan den sparas för att undvika att falla under GDPR.

Under GDPR så är entiteten, som bestämmer ändamålet för persondatan, ansvarig för bland annat att datan hanteras lagligt, bidra med information om rättigheter och skyldigheter samt hantera datan på ett säkert sätt och skydda den från intrång [11].

2.1.7 Etik

För att avgöra om något är etiskt bedöms det ur ett flertal kategorier, och de som har bedömts relevanta till detta arbete är yrkesetik, pliktetik och konsekvensetik. Yrkesetiken innebär att en utövare av ett yrke ska agera på ett sätt som anses moraliskt korrekt. I fallet med detta projekt innebär det att kod inte ska vara skadlig för en användare eller medföra säkerhetsrisker, framför allt när en produkt lanseras till kund.

Pliktetik är allmänna regler som varje människa har en skyldighet att följa. Exempel på dessa regler är lagar och generella värderingar som att alla ska behandlas med värdighet och respekt.

Till skillnad från de två tidigare kategorierna så tittar man endast på följderna av en handling i konsekvensetiken, där man försöker väga alla konsekvenser av en handling för att avgöra om en handling medför mer skada än nytta [12].

2.2.8 Hållbarhet

Precis som etiken kan hållbarheten delas upp i underkategorier där man får väga vilken hållbarhet som väger tyngst. Ett projekt kan bedömas som starkt hållbart om alla kategorier anses hållbara och varken det mänskliga eller naturliga kapitalet minskar. Även om stark hållbarhet inte kan finnas så kan det bedömas hållbart ur ett svagt perspektiv om summan av de båda kapitalen anses öka.

De aspekter som anses mest relevanta är den sociala och ekonomiska hållbarheten. Ekologisk hållbarhet anses ej relevant då applikationen ej förväntas bidra till en ökad användning av mobiltelefoner samt den ökade användningen av elektricitet anses försumbar.

Social hållbarhet delas upp i två relationer: horisontell och vertikal. För arbetet anses endast den horisontella relationen vara relevant då de påverkade parterna ej anses ha en maktbalans. I en horisontell relation så betraktas relationen mellan två likvärdiga individer, ur ett maktperspektiv, exempelvis två vänner.

Ekonomiska aspekter berör de två olika kapitalen. Det naturliga kapitalet, som är naturresurser som metaller och skog, samt mänskligt skapat kapital, vilket är allt från kunskap till infrastruktur. Ekonomisk hållbarhet berör hur dessa resurser behandlas och tas tillvara på nu och för framtiden [13].

2.2 Mjukvara

För att utveckla en mjukvara så krävs det ett antal verktyg som ofta är specifika till för den typen av lösning som eftersträvas. Enligt problemdefinitionen så ska lösningen fungera på två huvudsakliga plattformar: Android och iOS, därmed så har de olika alternativen av verktyg minskat avsevärt. De verktyg som har valts valdes huvudsakligen för att de förkunskaper som behövs för att använda dem redan finns och att de är gratis. På grund av dessa kriterier så kommer inte alternativa val att jämföras såvida det inte anses relevant till den slutprodukt som uppnås eller vid eventuell vidareutveckling.

2.2.1 Dart

Dart är ett programmeringsspråk utvecklat av Google och släpptes i början av 2010-talet. Språket används ofta för utvecklande av mobilapplikationer då den kan kompilera till både ARM64 och x86 samt Javascrip för web. Detta möjliggör att kod enbart behöver skrivas en gång och kan kompileras till flera plattformar. Dart använder just-in-time kompilering vilket effektiviserar utvecklingen då hela koden inte behöver kompileras på nytt vid en modifikation i koden vilket möjliggör snabb kompilering vid varje sparande av filen [14]. Variabler är typsäkra så de kräver en typdeklaration men språket har stöd för dynamiska typer som möjliggör mer flexibla variabler [15]. Strukturen är klassbaserad och uppbyggd av djupt nästlade klasser.

2.2.2 Flutter

För att utveckla mobila applikationer till Android med hjälp av dart så tog Google fram ramverket Sky, som senare blev Flutter. Ramverket är centrerat runt programspråket Dart och använder Darts klassbaserade struktur för att bygga Widgets. Widgets underlättar återanvändandet av kod som redan har skrivits samt tillåter importen av paket med funktioner som utomstående utvecklare har publicerat. Båda de nämnda funktionerna minskar utvecklingstiden som krävs vid utvecklandet av stora applikationer.

2.2.3 Paket

För att enkelt kunna återanvända eller dela kod så använder Flutter sig av paket som oftast består av widgets, hjälpfunktioner eller plugins. Möjligheten att ta in kod skriven av andra utvecklare eller för andra projekt kan spara på utvecklingstiden avsevärt samt underlätta för uppdatering av funktionalitet samt underhålla koden [16].

2.2.4 Android Studio

Android Studio är en simulator och IDE som är skapad av Google och Jetbrains för att vara den officiella IDE:n och utvecklingsmiljö för Android utveckling. Programmet har en inbyggd simulator för att simulera Android enheter och köra applikationer på enheterna. Simulatorn har möjlighet att köra extern kod och bra integration med Flutter [17].

2.2.5 Visual Studio Code

Valet av utvecklingsverktyg är ett väldigt personligt val och görs ofta baserat på vilket verktyg som utvecklaren är bekväm med. Visual Studio Code är en utvecklingsplattform som kan köras på Windows, Linux och Mac samt är öppen källkod. Plattformen är centrerad kring mappbaserad utveckling vilket underlättar vid utvecklingen av klassbaserade mobilapplikationer. För att arbeta med ett projekt så installerar man extensions från Visual Studio Code stora bibliotek som tillåter enkel installation av SDKs (Software Development Kit) samt sömlös integration av git hantering [18].

2.2.6 Git/Github

Hantering av versioner är en central del av utvecklingen när flera utvecklare arbetar på samma mjukvara. Git är en versionshanterare med stöd för många av de viktigaste funktionerna för programutveckling. Funktioner som att rulla tillbaka mjukvaran till gamla versioner, se ändringshistorik för individuella filer, se skillnader mellan olika versioner och mycket mera. Github är en plattform för att kunna spara, planera och spåra ett programmets utvecklingshistorik med hjälp av git. Plattformen underlättar samarbete mellan utvecklare med ett flertal verktyg för att hantera saker som kollisioner under sammanslagning av kod samt planering av utvecklande [19].

2.2.7 Docker

För att försäkra sig om att en användare eller utvecklare kan köra den mjukvara som skickas ut kan man använda Docker. En lokalt utvecklad mjukvara kan med säkerhet köras oavsett plats och användare med hjälp av containrar som kapslar in en applikations alla beroenden. En container skiljer sig från en virtual machine (VM) genom att köra program direkt på målets operativsystem och kan därmed startas på en väldigt kort period i jämförelse med en VM [20].

3 Metod

Projektets utveckling kan delas upp i tre delar: Utvecklandet av en kalenderapplikation, utforskande av metoder för ökad användbarhet, och implementering och utvärdering av användbarhetsökande metoder.

3.1 Utvecklandet av kalenderapplikationen

Utvecklandet av testplattformen utförs utefter det planeringsschema som tagits fram för att projektet skulle godkännas för genomförande. Under tiden som projektplanen utförs och implementeras så behöver planen förändras då vissa delmoment tar kortare eller längre tid att implementera än förväntat. När plattformen är färdigkonstruerad så implementeras de lärdomar som framkommer under efterforskningen.

3.1.1 Planeringsschema

Innan projektets start så skapades en utvecklingsplanering, se bilaga 1, där utvecklingsprocessen delades upp över tio veckor. Planeringen tog hänsyn till den tid som skulle krävas för att utveckla en kalenderapplikation som under tiden kunde testas med användbarhetsökande metoder. I slutet av den planerade tiden så finns en fungerande applikation som kan agera som en testplattform, inte en färdig produkt. Som kan ses i schemat så finns det tre stora milstolpar under utvecklingstiden där applikationen är i ett körbart tillstånd med alla påbörjade funktioner färdigställda.

3.1.2 Iteration och Reflektion

Under utvecklingstiden så sker det regelbunden reflektion av vad som har implementerats och förbättringsmöjligheter dokumenteras och katalogiseras i en lista som rankas efter varje punkts prioritet.

I slutet av varje utvecklingssteg så ska listan ha implementerats, senarelagts eller kasserats. För att en idé ska senareläggas så ska den bedömas att vara beroende av ett senare moment eller kräva ytterligare utvecklingstid. Om en idé kasseras så krävs det att den inte anses vara väsentlig för projektets ändamål, att öka användbarheten, eller kräva för lång utvecklingstid för att implementeras.

3.1.3 Utvecklingssteg

Första steget av utvecklingen är att skapa ett nytt Flutter projekt i Visual Studio Code med flutterkommandon i terminalen. I projektet skapas ett kodskelett som kan sedan simuleras i android studio för att försäkra att alla beroende paket fungerar så att utvecklingen kan fortsätta utan grundläggande problem.

När projektet kan simuleras utvecklas de mest grundläggande funktionerna enligt målen för en grundläggande kalender. De första funktioner som implementeras är ett schema för dagens datum och en metod för att lägga till och visa event under dagen. Schemats event hanteras av en central eventhanterare. Dessa funktioner expanderas sedan för att visa ett schema för den nuvarande veckan och hantera event under samma tidsperiod. När en vecka kan hanteras så vidareutvecklas det till vilken vecka som helst inom en tioårsperiod.

Med alla grundläggande schemavyer tillagda och en enkel event skapare implementerad påbörjas skapandet av en avancerad meny. Denna meny ska möjliggöra en mer avancerad planering med fler parametrar. Den enklare menyn får en algoritm för att hjälpa till att planera utan ytterligare information som behövs vid avancerad planering. Algoritmen är en grundläggande algoritm som planerar allt i första hålet den hittar i schemat. Denna grundläggande algoritm kommer att jämföras och ersättas med andra algoritmer.

Efter hand som efterforskningen hittar nya och bättre lösningar på problem så läggs de till i listan med förbättringsmöjligheter med önskade funktioner. Listan implementeras löpande och innehåller både nya idéer från efterforskningen och de som har kommit fram under regelbundna reflektioner.

3.2 Utforskande av metoder för ökande av användbarhet

För att hitta ytterligare funktionalitet att implementera så görs en kontinuerlig litteraturundersökning. Resultaten av undersökningen förs fram av utvecklarna under reflektionstiden efter en veckas leveranser och diskuteras. För att en användbarhetsökande funktion ska implementeras så sammanvägs ett antal faktorer däribland implementations tid, bidragande till diskussion, jämförbarhet och liknelse till andra implementationer.

Den tid det tar att implementera en funktion behöver vara sådan att den inte försenar andra delar av projektet samt hinner själv implementeras under projektets gång. Detta vägs ofta mot fördelarna av de andra kategorierna och är ofta den mest begränsande faktorn. Om en funktion inte kan bidra till diskussionen utan faller utanför målet för rapporten, exempelvis att den är riktad mot fel målgrupp, så kommer den att ej implementeras och vidare efterforskning anses ligga utanför projektets ram.

I det fall att en funktion bedöms som intressant för projektet och implementerbar inom en rimlig tid så kan den fortfarande nekas implementation om den anses för lik en redan implementerad funktion, exempelvis två funktioner som är algoritm beroende, då lärdomarna från en funktion kan implementeras för den andra. Funktionen kan även nekas implementation om den inte kan jämföras med alternativa lösningar. En studie som jämför en implementation med saknaden av en sådan ofta resulterar i ett binärt svar som kunde resonerats fram utan implementation.

3.3 Utvärdering av användbarhet

Med hjälp av den tidigare etablerade utvärderingsmetoden, Nielsens heuristiska utvärdering, utformas en testmiljö samt process som utförs och utvärderas av testare. Testningsprocessen är utformad i linje med Nielsens metod. Därmed kommer användarna först att ges en kort introduktion på hur testmiljön fungerar, därefter kommer handläggaren att ge dem de uppgifterna som ska utföras. Efter varje uppgift svarar testaren på ett antal utvärderingspunkter. Om en testare behöver hjälp i ett steg så kan handläggaren hjälpa till och sedan dokumentera att hjälp behövdes.

3.3.1 Testmiljö

Under testets utförande så kommer en handläggare starta applikationen på en laptop och simulera en telefon i android studio. Applikationen innehåller ett schema med ett antal event för att simulera en devis planerad vecka. När uppgiften avslutas av testaren blir den presenterad med verbala frågor som testledaren fyller i svaren på i Google Sheets. Efter att formulären har fyllts i så öppnar handläggaren en ny version av applikationen med ett nytt schema, där nya uppgifter utförs och utvärderas. Totalt visas fyra olika versioner av applikationen under testet som tillsammans representerar alla olika funktioner.

3.3.2 Utformning av uppgifter

Uppgifterna reflekterar användningsscenarier som förväntas uppstå vid normal användning av en funktion. För att minska variationen i jämförelsen så görs samma testuppgifter för alla olika implementationer. Uppgifterna som kommer utföras av testarna är följande:

1. Du ska ha ett samtal med en person som heter Kalle klockan 12.15 på måndag och samtalet tar 90 minuter. Planera in mötet i den digitala kalendern.
2. Du har ett par ärenden som behöver planeras in i ditt schema. Låt kalendern schemalägga ärendena åt dig (Försök att få ärendena på en rimlig tid också).
    a. Du vill dammsuga och vet att det tar 70 minuter att dammsuga huset
    b. Du vill även tömma diskmaskinen som tar 10 minuter
    c. Samt åka till gymmet och träna som tar 90 minuter
3. Kalle vill ut och spela discgolf med dig och har bett dig använda din fina nya app att planera när ni ska göra det. Kalle är tillgänglig tisdag till fredag och det brukar ta 3 timmar och 40 minuter.
4. Din kompis Nils kommer till stan mellan tisdag och fredag och vill dricka öl, du vill att din applikation ska planera in den dag och tid som ni ska till puben. Ni vill dricka öl i 4 timmar och 30 minuter.

3.3.3 Utvärderingspunkter

Tre olika typer av funktioner ska utvärderas enligt de punkter som Nielsen tagit fram, som anses relevanta. Olika punkter bedöms relevanta till olika funktioner, därmed skapas tre separata samlingar med frågor som berör olika punkter utefter vad som utvärderas. Dessa samlingar är algoritmfunktioner, menyfunktioner och inmatningsfunktioner.

De punkter som anses relevanta till funktionen utvärderas med hjälp av frågorna nedan(pN denominerar vilken av Nielsens heuristiska punkt frågan berör).

Algoritm

- p1: Hur väl förstår du hur algoritmen har tänkt?
- p2 : Hur väl motsvarar alternativen som applikationen presenterade dig gentemot det du hade förväntat dig om du bad en vän planera åt dig.
- p3: Hur begränsad känner du dig av funktionen? Är det enkelt att avbryta, ändra saker? Var det lätt att hitta?
- p5: Hur lätt var det att göra fel?
- p10: Hur mycket hjälp behövde du från handledaren?

Inmatning

- p2: Hur väl efterliknar inmatningen det du hade förväntat dig?
- p3: Hur lätt var det att avbryta en inmatning eller ändra klockslag under inmatningen?
- p5: Hur lätt var det att göra fel?
- p7: Hur mycket snabbare hade du kunnat utföra uppgiften med mer erfarenhet?
- p9: Om du gjorde fel, hur lätt var det att förstå vad som blev fel?
- p10: Hur mycket hjälp behövde du från handledaren?

Menyer

- p1: Hur lätt var det att förstå om en algoritm används eller inte?
- p2: Hur väl stämde mängden information som du behövde mata in med den mängd information du tycker man behöver för att planera samma sak?
- p5: Hur lätt var det att göra fel?
- p6: Hur mycket information behövde du komma ihåg från tidigare menyer?
- p7: Hur mycket känner du att skicklighet påverkade din planering hastighet?
- p8: Hur många alternativ kände du var överflödiga ditt ändamål?
- p9: Om du gjorde fel, hur lätt var det att förstå vad som blev fel?
- p10: Hur mycket hjälp behövde du från handledaren?

4 Systemkonstruktion

LatPlan har två huvudsakliga komponenter: en grundläggande applikation som fungerar som en testplattform för de användbarhetsökande metoderna, och de metoder som testats. Metoderna grupperas enligt de tre kategorier som utveckling har fokuserat runt, kategorierna: Menyer, Inmatningssystem och Algoritmer, innehåller alla ett flertal olika implementationer som senare kommer senare att jämföras och utvärderas gentemot varandra.

4.1 Grundläggande applikation

För att möjliggöra testandet av de användbarhetsökande funktioner som framtagits, har en grundläggande applikation skapats.

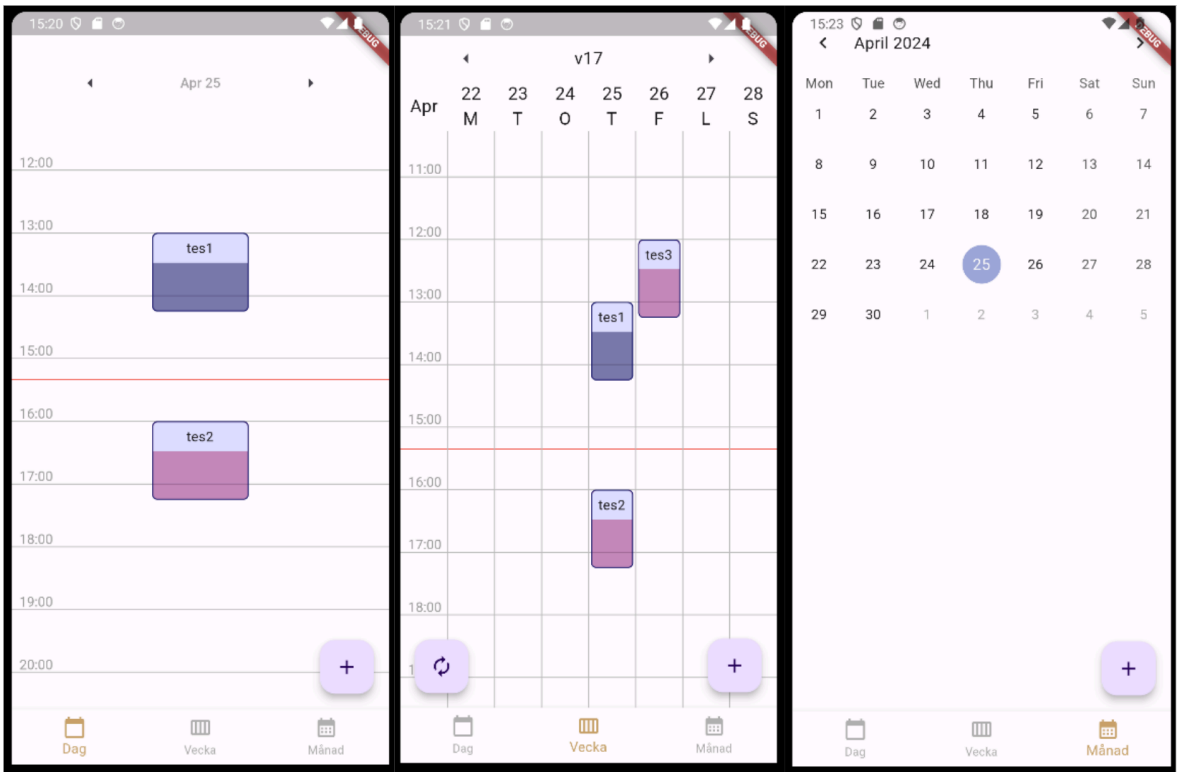


Bild 4.1: LatPlan i dagsvy, veckovy samt månadsvy.

LatPlan består av tre vyer som användaren kan växla mellan med hjälp av en meny vid botten av skärmen. Vyerna kan också växlas mellan om användaren svajpar till respektive vy. Dessa vyer är de olika schema-typerna som implementerats och de består av dagsschema, veckoschema, samt månadsschema som kan ses i bild 1. I schemat finns också en röd linje som visar den nuvarande tiden för användaren. Varje schema har också en meny vid toppen av skärmen där användaren kan se information om schemat som visas.

Dagsvyn består av ett schema för en specifik dag som kan väljas med knapparna i den övre menyn eller med månadsvyn. Schemat visar eventen som infaller på den dagen och användaren kan scrolla upp och ner för att se alla tider på dagen.

Veckovyn har ett schema som bygger på dagsschemat men har utökats för att visa schemat sju dagar samtidigt. knapparna i den övre menyn byter vilken vecka som visas och visualiserar information om veckonummer samt alla datum på veckans dagar.

Knappen nere i vänstra hörnet på veckovyn schemalägger hela schemat på nytt med hjälp av en algoritm för att skapa ett mer optimerat schema. Det är enbart event som faller inom en tredagars period efter den nuvarande dagen som påverkas.

Varje event i schemat faller inom en av två kategorier, det är antingen ett statiskt eller ett löst planerat event. Statiska event kommer inte att flyttas på när hela schemat schemaläggs på nytt, medan de löst planerade eventen kan schemaläggas på nytt vid en total schemaläggning.

Den sista vyn är månadsvyn som består av ett månadsschema där användaren kan se om det infaller några event vid en specifik dag. Användaren kan också interagera med schemat för att ta sig till dagsvyn för den dagen som klickades på. Detta ändrar också veckoschemat till den vecka som den valda dagen faller inom. Vid toppen av skärmen kan användaren också växla månad för schemat.

Bild 4.2: Event meny för att modifiera event

Alla event i både dagsschemat och veckoschemat går att interagera med för att ändra eventets uppgifter. Här kan eventets titel, datum, och status ändras som kan ses i bild 2.

4.2 Menyer

För att en användare enkelt ska kunna navigera de alternativ som finns för inmatning har ett menysystem implementerats. För att hantera flera olika typer av inmatningsdata så har en avancerad meny i samband med en förenklad meny implementerats. För att möjliggöra en utvärdering av implementationen så skapades även en kombinerad meny där alla alternativ som finns i de två menyerna lades i en meny, så kallad kombinerad meny,

4.2.1 Förenklad Meny

En förenklad meny är ett koncept som tillåter en användare att endast se de inmatningsfält som är väsentliga för att en funktion ska kunna utföras, med mål att förbättra användbarheten vid enkla inmatningar. All utökad funktionalitet är undangömd vilket kan bidra till en upplevd försämring av användbarheten, i form av fler knapptryckningar eller en svårhittad funktionalitet vid mer avancerade operationer.

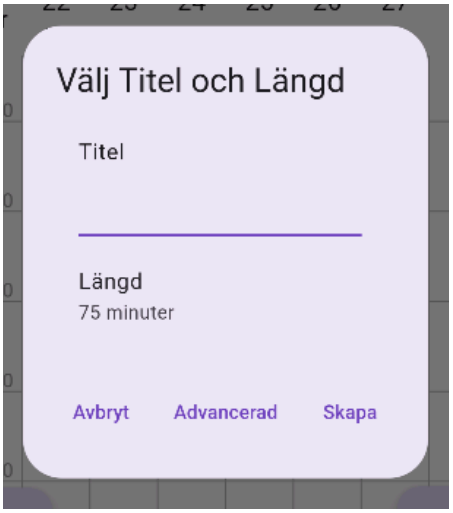


Bild 4.3: Förenklad meny för skapande av event.

Den förenklade menyn som har implementerats har två inmatningsfält, som kan ses i bild 3. Menyn tillåter inmatning av en titel och en längd på eventet. Detta görs möjligt genom en planeringsalgoritm som planerar eventet åt användaren inom ett standardintervall på tre dagar med start på morgondagen.

4.2.2 Avancerad Meny

För att komplettera den förenklade menyn implementeras en avancerad meny som lätt kan nås från den förenklade menyn. Den avancerade menyn är till för att tillåta användande av de funktioner som har gömts undan för att skapa den förenklade menyn.

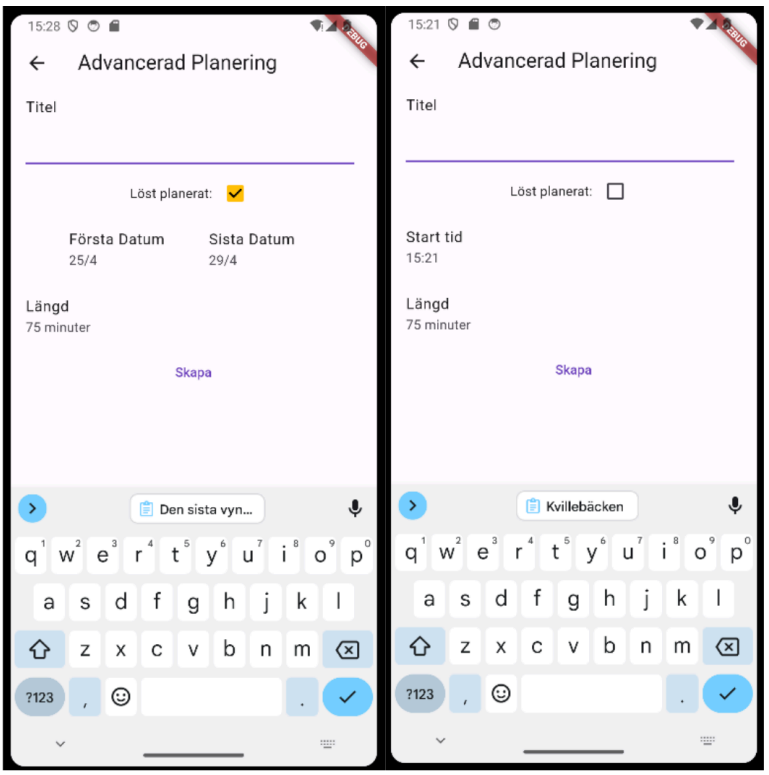


Bild 4.4: Avancerad meny för att lägga till två event

Implementationen av en avancerad meny som kan ses i bild 4 kan lätt nås från den förenklade menyn med hjälp av “avancerat” knappen som kan ses i bild 3. När en användare väljer att göra en avancerad planering så möts de av ett flertal alternativ. Alternativen låter en användare välja mellan att planera in ett event på en specifik tid eller specificera över vilket spann som algoritmen ska planera åt en.

4.2.3 Kombinerad Meny

Denna meny konstruerades med avsikten att hantera all funktionalitet på ett ställe. Detta tillåter en användare att direkt nå den funktion som hen önskar att använda.

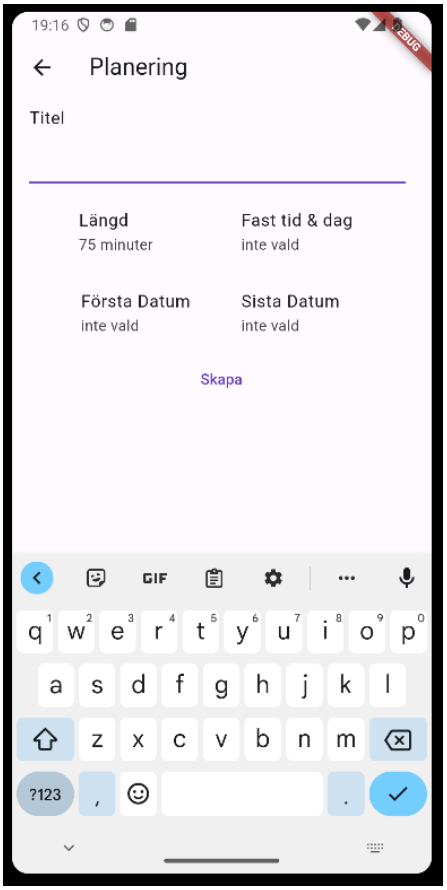


Bild 4.5: En kombinerad meny för att lägga till event

Som kan ses i bild 4.5 så presenteras all funktionalitet för användaren och kan användas direkt. Applikationen läser in den data som angivits i fälten och beroende på vilka fält som använts bedömer applikationen om en algoritm bör hantera informationen, eller om det finns tillräckligt med information för att direkt schemalägga ett event utan algoritm. Fälten korrigeras av applikationen om en användare har använt fält som är inkompatibla med varandra.

4.3 Inmatningssystem

Under informationssökning så har många olika implementationer av inmatningssystem hittats och tre olika inmatningssystem som ansågs passande för ändamålet, inhämtandet av en tid, samt tillräckligt olika har implementerats. De metoder som har valts är ren tangentbordsinmatning, en representation av en analog klocka samt en rullande digital klocka.

4.3.1 Tangentbordsinmatning

Den mest universella inmatningsmetoden, tangentbordsinmatning, låter användaren skriva in text i en ruta med hjälp av telefonens inbyggda digitala tangentbord. För att underlätta för användaren att skriva tiden i rätt format så har den tillåtna inmatningen begränsats till siffror.

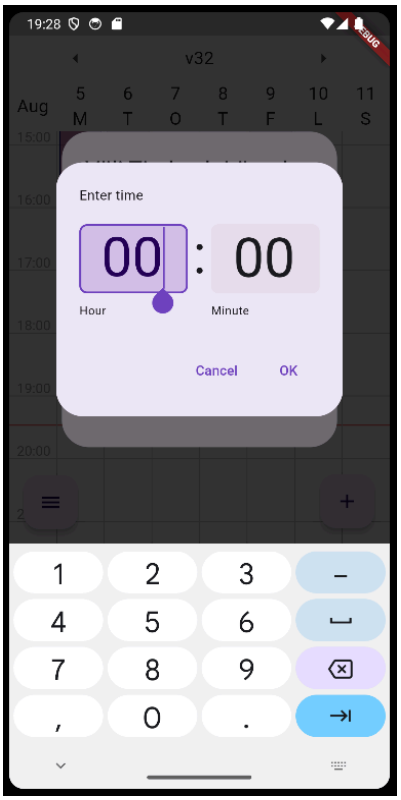


Bild 4.6: Tangentbordsinmatningsruta med tangentbord

I bakgrunden så presenteras inmatningen som en digital klocka med två inmatningsfält, en för timmar och en för minuter, som kan ses i bild 4.6. Om en användare försöker mata in en längd med ett minut värde större än 59 så förändras det av applikationen till 59 minuter.

4.3.2 Analog klocka

Flutters egna inmatningssystem är i form av en representation av en analog klocka där en användare drar med fingret till den mängden timmar som önskas och sedan upprepar för minuter.

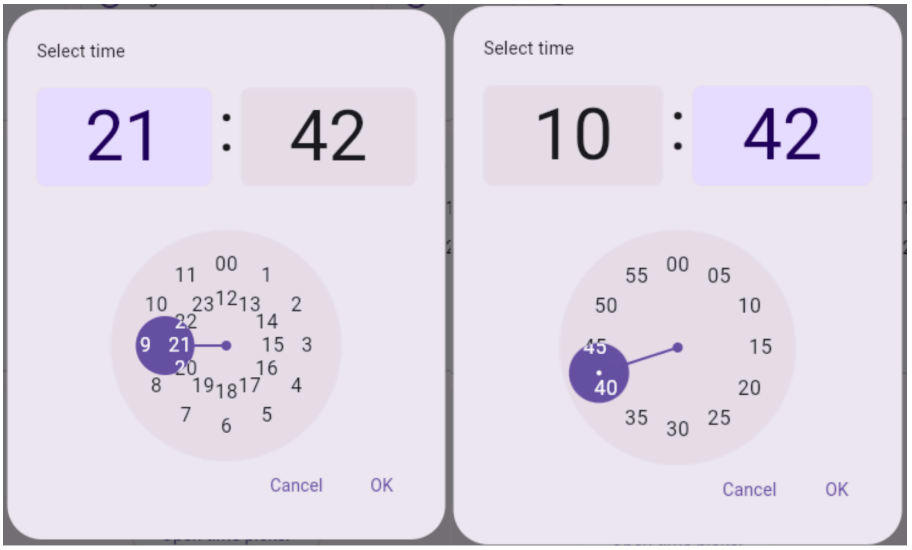


Bild 4.7: Analog klocka för tids inmatning vid val av timmar respektive minuter

Inmatningen sker med hjälp av två klockor där första klockan anger timmen och den andra anger minuten som kan ses i Bild 4.7. Implementationen lämnar lite utrymme för inmatning utanför tillåtna intervaller men begränsar antalet timmar till 24.

4.3.3 Rullbar digital klocka

iOS använder en tidsinmatningsmetod där användaren väljer tid genom att rulla på två hjul som respektive anger timme och minut. Eftersom denna inmatningsmetod inte förekommer i standardbiblioteket för flutter installerades ett externt paket med denna metod.

Bild 4.8: Rullbar digital klocka

Tidsväljaren består av två hjul som användaren rullar på, som kan ses i bild 4.8, för att välja rätt tid. Implementationen bygger på att användaren känner igen att den representerar en digital klocka då den saknar rubriker för rullarna.

4.4 Algoritmer

För att tillåta automatisk planering samt begränsa de alternativ som användaren behöver ange så implementerades en algoritm. För att undersöka om en algoritm kan bidra till ökad användbarhet så har flera olika algoritmer implementerats och testats.

4.4.1 Första bästa tid

Den första algoritmen som implementerades var en grundläggande algoritm som planerar in ett event på den första tiden den får plats på som inte kolliderar med ett annat event. Algoritmen tar endast hänsyn till den tid som ett event tar upp och inga andra faktorer. Ett event planeras in på morgondagen och algoritmen försöker passa in eventet från klockan åtta på morgonen. Vid en kollision går algoritmen vidare i schemat och testar en senare tid. Detta upprepas till en tid där eventen får plats utan kollision hittats.

4.4.2 Slumpad

För att undersöka den motsatta extremen av algoritmer så implementeras en slumpad algoritm som genererar tre slumpvis utvalda tider som inte kolliderar med andra event som en användare kan välja mellan. Algoritmen genererar slumpvist fram tider för eventen, där varje tid avrundas till närmaste femte minut. Tiderna som genereras förekommer enbart mellan morgondagen och midnatt tre dagar senare.

Möjligheten att välja mellan tre olika alternativ gör att en användare kan välja det alternativ som passar bäst. Problematik kommer att uppstå när alla tre genererade alternativ är opassande och föreslås vid dåliga tider.

4.4.3 Slumpad med vikt utan inlärning

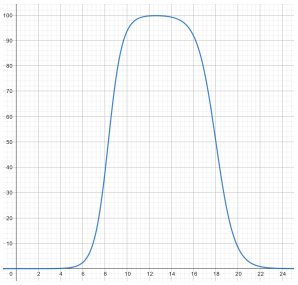
För att algoritmen ska ta fram mer frekventa förslag på lämpliga tider som tar hänsyn till aspekter inom det nuvarande schemat så utvecklas den existerande slumpgenererade algoritmen. Det genereras 100 stycken permutationer av schemat där det nya eventet schemaläggs med den tidigare slumpade algoritmen. Sedan utvärderas alla permutationer och betygsätts i tre kategorier med ett värde mellan 0 och 100 i varje kategori. Dessa värden summeras sedan till en totalpoäng för varje schema mellan 0 och 300. Kategorierna som utvärderas är tid på dygnet, tid mellan event, och fördelning av event. Användaren blir sedan presenterad med de tre genererade tiderna med högst poäng och kan sedan välja det som passar bäst.

För att utvärdera permutationerna, i de olika kategorierna, så används matematiska funktioner som poängsätter utifrån schemats resultat i förhållande till ett idealiskt värde. Alla eventen inom planeringsspannet evalueras vilket inkluderar det nyligen tillagda eventet. Den första kategorin utvärderar schemat utifrån vilka tider på dygnet alla event är schemalagda på. Eventen betygsätts enligt formeln där starttiden är x.

$maxPoäng = 100$

$stigandeGrad = 1,6;$ $\quad fallandeGrad = 1,2$

$tidigast = 8,3;$ $\quad senast = 18$



Figur 4.1: p(x) i en graf

$$p(x) = \frac{maxPoäng}{1 + e^{stigandeGrad(tidigast - x)} + e^{fallandeGrad(x - senast)}}$$

Denna formel är framtagen för att event schemalagda mellan 10.00 och 16.00 ska bli tilldelade över 90% av maxpoäng. Event som faller före 10.00 och efter 16.00 blir tilldelade mycket lägre poäng som kan ses i figur 1, där poängen är en funktion av den angivna starttiden för ett event. Formeln är uppbyggd av två delfunktioner som skapar den stigande respektive den fallande kurvan.

$$medelstarttid = \frac{1}{n}\sum_{i=1}^{n} p(x_i)$$

Där n är antalet event och x är starttiden för ett specifikt event. Varje schema betygsätts i helhet och inte för endast det tillagda eventet för att möjliggöra planeringen av en samling av event. Funktionen p(x) itereras med alla event i ett schema och ett medelvärde, kallat medelstarttid, framtas utifrån summan enligt ekvationen ovan. Detta ger ett medelvärde mellan 0 och 100.

Den andra kategorin som utvärderas är det genomsnittliga mellanrummet mellan alla event i schemat. Ett mellanrum definieras som den tid mellan två event som sker samma dag och inget event faller under tiden.

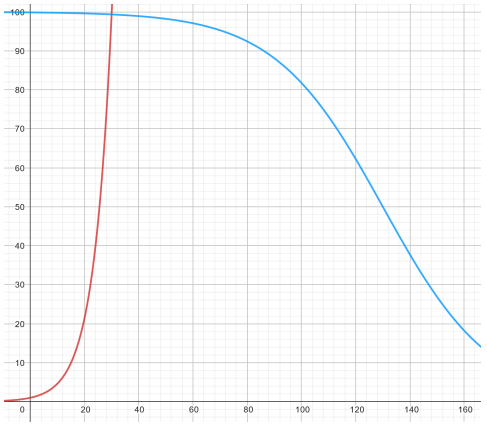
$$maxPoäng = 100$$

$$alpha = 30$$

$$fallandeGrad = 0,05$$

$$g(x) = e^{\frac{ln(maxPoäng)*x}{alpha}}$$

$$k(x) = \frac{maxPoäng}{1+e^{fallandeGrad*(x-maxPoäng-alpha)}}$$



Figur 4.2: g(x) i röd samt k(x) i blått i en graf

Denna formel är framtagen för att scheman där tiden mellan eventen är minst 30 minuter ska bli tilldelade hög poäng. Scheman där eventen med extra mellanrum prioriteras också för att ytterligare ett event ska få plats. Som kan ses i figur 2 så straffas mellanrum som är för små avsevärt mer än större mellanrum. Detta är för att en användare alltid antas vilja ha ett minsta mellanrum för transport eller förberedelser.

Den tredje kategorin som värderas är spridningen av event mellan dagarna. Detta enligt funktionen nedan där Δn är skillnaden av event mellan den dagen med flest antal event och den dagen med minst antal event.

$$maxPoäng = 100$$

$$beta = 2$$

$$förskjutning = 2,5$$

$$j(\Delta n) = \frac{maxPoäng}{1+e^{beta(\Delta n-förskjutning)}}$$

Målet med denna kategori är att skapa jämn spridning av event mellan dagarna medan fortfarande tillåta scheman med nödvändig spridning.

4.4.4 Slumpad med vikt och inlärning

Genom att implementera inlärning för den ovan nämnda algoritmen kan planeringen göras mer personlig. Den inlärning som implementerats använder sig av de val som en användare gör när den väljer en av de tre olika tider som presenteras av de tidigare algoritmerna. Information om det nya eventets starttid, avstånd till andra event, samt spridningen av event i det nya schemat används för att förbättra algoritmernas beslut i senare schema generering. Då varje kategori kräver sin egen form av inlärning har de tidigare formlerna modifierats för att använda var sin sparad variabel för att ändra deras beteende. Dessa variabler modifieras utefter informationen insamlad efter ett val med hjälp av tre nya funktioner.

Den första funktionen modifierades med ett förskjutningsvärde som ändrar de optimala timmarna för ett event där maxpoäng tilldelas. Då förskjutningsvärdet kan vara positivt eller negativt så kan de optimala timmarna falla tidigare eller senare på dagen.

$$maxPoäng = 100$$

$$stigandeGrad = 1,6$$

$$fallandeGrad = 1,2$$

$$tidigast = 8,3$$

$$senast = 18$$

$$p(x) = \frac{maxPoäng}{1 + e^{stigandeGrad(tidigast - x + förskjutning)} + e^{fallandeGrad(x - senast + förskjutning)}}$$

Förskjutningsvärdet modifieras beroende på medelvärdet av alla starttider i schemat efter att ett nytt event skapats av algoritmen. Då förskjutningsvärdet är beroende av sitt gamla värde för att skapa en mjukare förskjutning används en rekursiv formel med en konstant för att minska förändringshastigheten.

$$råförskjutning = medelstarttid - 12$$

$$mjukFactor = 0,05$$

$$förskjutning_0 = 0$$

$$förskjutning_{a+1} = förskjutning_a + (råförskjutning - förskjutning_a) * mjukFactor$$

Formeln bygger på förhållandet mellan den nuvarande förskjutningen i schemat och den tidigare sparande variabeln. Detta ger oss skillnaden på hur schemat är förskjutet jämfört med de tidigare antagna preferenserna i variabeln. Skillnaden är negativ om schemat har en medelstarttid tidigare än förväntat och positivt om schemat har en medelstarttid senare än förväntat.

Den andra funktionen använder sig utav samma modifiering med ett förskjutningvärde som antingen ökar eller minskar den önskade storleken på mellanrummen i schemat.

$$mjukFactor = 0,2$$

$$alpha_0 = 30$$

$$alpha_{a+1} = alpha_a + (medelMellanrum - alpha_a) * mjukFactor$$

$$maxPoäng = 100$$

$$fallandeGrad = 0,05$$

$$g(x) = e^{\frac{ln(maxPoäng)*x}{alpha}}$$

$$k(x) = \frac{maxPoäng}{1+e^{fallandeGrad*(x-maxPoäng-alpha)}}$$

I dessa formler är alpha det önskade mellanrummet mellan event i schemat. Alpha använder samma formel som den tidigare funktionen med ändrade konstanter. Funktionen har också ett högre värde på hur snabbt värdeförändringar sker, detta för att alpha behöver större förändringar i värde för att påverka g(x) och k(x).

$$mjukFactor = 0,05$$

$$beta_0 = 2$$

$$beta_{a+1} = beta_a + (spridning - beta_a) * mjukFactor$$

$$j(\Delta n) = \frac{100}{1+e^{beta(\Delta n-2,5)}}$$

Den tredje och sista funktionen använder beta som lutningsvärde i formeln. Eftersom beta kan vara negativt kan funktion växla från att prioritera schema med låg spridning till scheman med hög spridning. Om beta närmar sig noll kommer funktionens derivata att närma sig noll. Vilket gör att scheman med låg spridning och scheman med hög spridning blir tilldelade lika många poäng.

Även denna formel använder sig av samma formel som de andra funktionerna. Schemats spridning jämförs med den önskade spridningen och skillnaden används för att modifiera beta.

Skiftandet av de värden som anses optimala för betygsättning gör att de föreslagna tiderna kan anpassas efter varje användares behov. På grund av den nuvarande implementeringen så kommer en användare med en skillnad i önskemål gentemot startvärdet behöva använda LatPlan under en längre period innan värdena kommer att ha skiftats för att reflektera de önskade värdena.

5 Resultat

I kapitlet kommer en sammansättning av resultatet från den heuristiska utvärderingen presenteras. Resultatet är uppdelat i tre delar baserat på de tre olika områden som de användbarhetsökade metoderna har grupperats i tidigare.

5.1 Algoritmer

Tabell 5.1: *Sammanställning av det genomsnittliga betygen för algoritmer under heuristiska utvärdering 1 är sämsta möjliga resultat och 5 är bästa.*

	Första bästa	Slumpad	Viktade	Inlärning
Förstår du hur algoritmen har tänkt?	5	3,6	3,2	3,2
Hur väl motsvarar alternativen som applikationen presenterade dig gentemot det du hade förväntat dig om du bad en vän planera åt dig?	1,4	1,4	3,6	3,6
Hur begränsad känner du dig av funktionen? (Är det enkelt att avbryta, ändra saker?)	1,8	2,6	2,6	2,6
Hur lätt var det att göra fel?	5	4,6	4,6	4,4
Hur mycket hjälp behövde du från handledaren med algoritmen?	4,4	4,2	4	3,8

Efter testning med de första tre kandidaterna upptäcktes det att den viktade algoritmen med inlärning producerade samma scheman som den utan inlärning och därmed gav väldigt liknande svar. Många av testarna frågade om hjälp med vilken tid handledaren tyckte att de skulle välja för de kände stress att inte göra fel. Handledaren fick påminna att det inte fanns några felaktiga val och att testaren kunde välja det som känns mest naturligt.

5.2 Inmatningsmetoder

Tabell 5.2: *Sammanställning av det genomsnittliga betygen för inmatningsmetoder under heuristiska utvärdering 1 är sämsta möjliga resultat och 5 är bästa.*

	Tangentbords inmatning	Analog klocka	Rullbar digital klocka
Hur väl efterliknar inmatningen det du hade förväntat dig?	2	2,6	4,2
Hur lätt var det att avbryta en inmatning eller ändra klockslag under inmatningen?	4,6	4,4	4
Hur lätt var det att göra fel?	3,8	2	3,8
Hur mycket snabbare hade du kunnat utföra uppgiften med mer erfarenhet?	3	3	3
Om du gjorde fel, hur lätt var det att förstå vad som blev fel?	4,6	4,6	4
Hur mycket hjälp behövde du från handledaren med inmatningen?	4,8	3,8	5

Under testningen av den analoga klockan som inmatningsmetod så behövde en testare hjälp av testledaren, då hen inte förstod hur man angav en längd på ett event med hjälp av klockan.

5.3 Menyer

Tabell 5.3: *Sammanställning av det genomsnittliga betygen för menyer under heuristiska utvärdering 1 är sämsta möjliga resultat och 5 är bästa.*

	Kombinerad	Förenklad med avancerad
Hur lätt var det att förstå om en algoritm används eller inte?	2,2	2,8
Hur väl stämde mängden information som du behövde mata in med den mängd information du tycker man behöver för att planera samma sak?	4,2	5
Hur lätt var det att göra fel?	3,2	4,2
Hur ofta behövde du komma ihåg information från tidigare menyer?	3,6	4,6
Hur mycket snabbare hade du kunnat utföra uppgiften med mer erfarenhet?	3	2,4
Hur många alternativ kände du var överflödiga ditt ändamål?	4,6	4,8
Om du gjorde fel, hur lätt var det att förstå vad som blev fel	3,2	4,8
Hur mycket hjälp behövde du från handledaren med inmatningen?	3,2	4,6

Under testningen av den kombinerade menyn så behövde två testare hjälp av testledaren, med att förstå vilka fält som skulle fyllas i för att applikationen skulle planera åt testaren med hjälp av en algoritm.

6 Diskussion

I kapitlet kommer resultaten av den heuristiska utvärderingen diskuteras och metoden för utförandet av undersökningen kritiseras.Ytterligare aspekter som etik, hållbarhet och alternativa lösningar kommer att redogöras för att sedan rekommendera steg för att vidareutveckla projektet.

6.1 Heuristisk utvärdering

Resultaten av den heuristiska utvärderingen som har presenterats ger en helhetsbild av de olika metoder som har implementerats utifrån ett användbarhetsperspektiv. För att jämföra de olika lösningarna behöver varje fråga analyseras individuellt då olika frågor berör olika heuristiska punkter.

6.1.1 Algoritmer

Under undersökningen uppstod ett mönster bland algoritmerna. Testarna behövde sällan hjälp och tyckte det var svårt att göra fel. Detta speglas i att inget poäng medelvärde skiljer sig mer än 0.75 poäng från övriga algoritm implementationer bland dessa frågorna.

Man kan observera att “första bästa” algoritmen var lättast att förstå men att dess enkelhet bidrog till en frustration med den bristande planeringsfunktionaliteten vilket fick testarna att känna sig begränsade och frustrerade. Samma frustration med den bristande planeringsfunktionaliteten hittades även hos den rent slumpade algoritmen som däremot saknade samma förutsägbarhet. Denna frustration bidrar till en minskad användbarhet vid implementering.

Jämförelsevis ansåg testarna att den viktade med och utan inlärning planerade mer som människa hade planerat. Testarna kände sig mer begränsade av dem samt den rent slumpade algoritmen då fler val presenterades än vid användning av första bästa. Ingen markant skillnad mellan algoritmen med inlärning och den utan hittades av testarna. Därmed kan ingen ökad användbarhet visas vid implementering av inlärning.

Eftersom frågorna reflekterar Nielsens heuristiska punkter kan vi därmed konstatera att den funktion som anses bäst beror på vilka av punkterna utvecklaren värderar. Om den första punkten, synlighet av systemtillstånd, anses viktig nog att man kan helt bortse från den minskade användbarheten inom andra kategorier är “första bästa” den högst presterande. Däremot om man värderar punkt två, systemet och verkligheten, på samma magnitud som punkt ett så är de viktade algoritmerna bäst.

6.1.2 Inmatning

Resultatet av den heuristiska undersökningen konstaterar att testarna klart förväntar sig en digital klocka vid inmatning av en tid. Därmed har den digitalt rullbara klockan en betydligt bättre användbarhet enligt Nielsens andra punkt, systemet och verkligheten. Detta återspeglas i frågorna som testarna ställde till handledaren under testandet av den analoga klockan.

Den digitala rullbara klockan var ej märkbart sämre än de andra implementationerna inom de andra kategorierna förutom möjligheten att avbryta en inmatning. Därmed minskar inmatningsmetodens användbarhet enligt Nielsens punkt tre, användarkontroll och frihet. Problematiken inom detta område misstänks bero på avsaknaden av en knapp för att avbryta inmatningen. Om problematiken löses i formen av tillägget av en avbryt knapp så kan lösningen anses vara den klart bästa utan konkurrens.

6.1.3 Menyer

Den förenklade menyn används i kombination med en avancerad meny gav testarna högre betyg på alla frågor utom en där skillnaden var marginell. Av de åtta punkterna som utvärderades var det därmed endast en av Nielsens punkter där implementationen ej ansågs bäst. Punkten var flexibilitet och effektivitet av användare, som avser möjligheten för avancerade användare att snabbare kunna utföra uppgifter än en nybörjare. Skillnaden var endast 0.6 poäng och anses därmed inte stor nog att medföra en större effekt.

6.2 Utvärdering av undersökning

Undersökningen av användbarheten baserades på de principer som Nielsen tagit fram och utvecklat bedöma användbarhet. Undersökningen begränsades därför till fem deltagare eftersom Nielsen under tidigare studier konstaterade att bidragandet till resultatet av individuella testare minskar drastiskt efter tre och blir knappt märkbar efter fem. Vid observation av testresultatet, som finns i bilaga 2, så kan man notera att testernas resultat inte skiljer sig mycket från varandra vid majoriteten av frågorna. I det fallet att det skiljer sig markant så är det oftast endast en testare vilket ej påverkar genomsnittet i stor del. För vidare utveckling av individuella metoder kan det vara intressant att utföra mer utförliga studier för att identifiera en optimal implementation.

6.2.1 Minskande av tidskrav

Tid var en stor begränsande faktor i hur utförligt varje testare kunde utföra undersökningen. Då varje person behövde testa 9 funktioner där varje funktion ej kan testas individuellt. Om varje permutation av funktioner skulle testas skulle det leda till 24 tester där varje test kan ha upp till 10 punkter att bedöma. Detta skulle leda till 24 testversioner med totalt 240 frågor vilket tar för lång tid. För att begränsa utförandetiden för testet bedömdes det att mängden permutationer behövde minska samt antalet frågor.

Permutationerna som valdes gjorde att varje funktion testades minst en gång. Den begränsande faktorn var mängden algoritmer vilket gjorde det minsta antalet permutationer till fyra. Därefter bestämdes vilka av Nielsens punkter som ansågs relevanta till de olika funktionsgrupperna.

6.2.2 Val av Nielsens heuristiska punkter

Punkterna som användes begränsades till de punkter som kunde påverkas av skillnader mellan olika implementationer, men eftersom vissa aspekter förblir oförändrade mellan versioner valdes punkter bort. Punkterna som valdes borr för varje respektive metod grupp var:

- Algoritmer
    - Följdriktighet och standarder
    - Igenkänning över ihågkommande
    - Flexibilitet och effektivitet av användare
    - Utseende och minimalisk design
    - Hjälp användare känna igen, diagnostisera och återhämta sig från fel
- Inmatningssystem
    - Synlighet av systemtillstånd
    - Följdriktighet och standarder
    - Igenkänning över ihågkommande
    - Utseende och minimalisk design
- Menyer
    - Användarkontroll och frihet
    - Följdriktighet och standarder

6.2.3 Problematik med genomförande

På grund av att inga av de implementerade funktionerna kunde testas individuellt utan en implementering från de andra kategorierna uppstod problematik vid utvärderingen. Testarna hade problem att bedöma endast funktionen som frågan berörde och behövdes vid flera tillfällen påminnas att frågorna inte var en helhetsbedömning. Problematiken kan ha påverkat resultatet men detta kan inte bekräftas. På grund av att genomsnitt diskuteras bör inte skillnaden vara tillräckligt stor för att påverka slutsatser från undersökningen om testarna systematiskt har svarat fel på samma fråga.

Vid sammanställning av resultatet observeras det att resultaten för de viktade algoritmerna, med och utan inlärning, har snarlika resultat. Efter vidare analys insågs det att hur testschemat var konstruerat i kombination med de preferenser som var förprogrammerade för inlärningen bidrog till att båda implementationerna skapade snarlika scheman. Då inlärningens föredragna tider var tagna av tidigare inlagda event och andra hand valen hamnade på de platser som algoritmen utan inlärning också valt. Inlärningens huvudsakliga funktion är att anpassa algoritmen efter användarens preferenser men detta kräver ett konstant användande över en längre tid innan den är anpassad. En lösning hade varit att intervjua testarna innan och ge anpassade inlärningsvärden baserade på resultatet av intervjun. Alternativt kan en annan typ av studie över en längre tid utföras med mål att bedöma just denna funktion.

6.3 Alternativa användbarhetsökande lösningar

För att komplettera den automatiska schemaläggningen av event med hjälp av algoritmer så kan ett aktivt schema skapas. Det aktiva schemat upptäcker om det schema som planerats via algoritmen har blivit dåligt efter att en användare har planerat in egna fast planerade event och kan signalera detta. Därefter kan LatPlan schemalägga alla automatiskt planerade event på nytt för att på så sätt skapa ett bättre schema. Detta har inte implementerats då det ansågs svårt att jämföra med andra liknande implementationer utan att endast fokusera på algoritmen, som används för att avgöra om schemat är bra. Detta skulle ge en väldigt liknande studie och resultat som utforskandet av planeringsalgoritmen.

En stor del av ökandet av användbarheten är att ta bort funktioner som användaren upplever som irriterande, men alla användare gillar inte samma saker. Därmed är den bästa bedömaren av vad användaren vill ha, användaren själv. För att tillåta användaren att bestämma själv vad som ska finnas kan en meny för att stänga av funktionalitet implementeras. Detta anses vara en bra användbarhetsökande metod som inte implementeras men varmt rekommenderas att undersökas vid utveckling av applikationer. Funktionen implementerades inte för att den ansågs riktas mot avancerade användare och inte har alternativa lösningar att utforska utan att implementera avsevärt olika grundapplikationer.

6.5 Etik

Projektet kan anses etiskt försvarbart då ingen av koden som utvecklas kan användas i ett skadande syfte, vilket uppfyller kraven för att vara försvarbar ur ett yrkesetiskt perspektiv, samt inte bryter mot några lagar eller sociala regler, vilket uppfyller kraven för pliktetik.

Diskussionen uppstår kring vidareutveckling då lagar som GDPR blir relevanta och måste följas. Funktioner som bygger på sparande av data centralt samt skapande av konton ställer ytterligare krav från applikationen. Detta medför att vissa rekommendationer som görs i rapporten kring funktioner som inte implementerats måste undersökas och implementeras på ett sätt som följer lokala lagar.

6.6 Hållbarhet

För att bedöma arbetets hållbarhet så analyseras projektet ur de två relevanta perspektiven. Projektets ambition ligger mycket i linje med social hållbarhet då ett fokus har legat på det användbarhetsökande arbetet för att underlätta användandet för folk med svårigheter att planera. Att underlätta för individer att delta i vårt samhälle i form av ökade sociala möjligheter eller minskad individuell bördan faller inom en stark hållbarhet ur ett socialt perspektiv.

När projektet betraktas ur ett ekonomiskt hållbarhetsperspektiv kan man härleda hur förbättrade planerings möjligheter leda till ökad produktivitet. Det finns inga naturliga resurser som skulle förminskas av LatPlans användande i ett testnings syfte, men vid användande av en större kundbas kommer det att tillkomma byggnation av infrastruktur som serverhallar. I projektets nuvarande ram så kan projektet anses ej minska det naturliga och öka det mänskliga kapitalet och därmed ha en stark hållbarhet.

6.7 Vidareutveckling

Implementationen av LatPlan följer den plan som konstruerades i början av projektet. Detta leder till att viss funktionalitet som ansågs vara viktig i ett tidigt skede har implementerats men kan nu konstateras ha tagit upp utvecklingstiden som hade kunnat användas för viktigare moment. Exempel som notifikationer implementerades för att uppfylla funktionskrav som skapades innan utveckling startade.

Notifikationer anses överflödiga då det inte har någon påverkan på LatPlans kapabilitet som en testplattform. De vyer som krävs för att kunna utvärdera resultaten anses vara begränsad till veckovy då den gav samma information som en dags vy och en månadsvy gav ingen ytterligare information som användes vid testerna. En stor del av utvecklingstiden lades på att få LatPlan att se visuellt trevlig ut, en användarvänlig design är väsentlig för en bra upplevelse, men var inte något som undersöktes och därmed inte nödvändigtvis behövdes implementeras.

Med LatPlan så har möjligheter för vidareutveckling skapats, för att underlätta dessa så har Docker-filer implementerats. Om applikationen ska vidareutvecklas till ett stadie där den kan anses redo för en kund så behöver fem huvudfrågor hanteras: Utveckling av användbarhetsökande funktioner, personinformation, säkerhet, kundundersökning och feedback. Områden berör de saker som har avgränsats från projektet på grund av LatPlans mål, att vara en testplattform för undersökandet av användbarhetsökande metoder.

För vidareutveckling så rekommenderas det att använda Android Studio till skillnad från Visual Studio Code som användes i projektet. Utvecklingsmiljön valdes på grund av att utvecklarna hade tidigare erfarenhet och plattformens Git integration. Men under projektets gång så noterades problem under simuleringen av applikationen som hade kunnat undvikas vid användande av Android Studio.

Projektet har endast jämfört skillnaden mellan användbarhetsökande lösningar. Därmed rekommenderas det starkt att en vidareutveckling, för en applikation för konsumenter, gör en kundundersökning och samlar in feedback kontinuerligt för att utvärdera vad för funktioner som är viktiga för konsumenten.

För att LatPlan ska tillåtas att publiceras på marknaden, och att arbetet ska fortsätta kunna anses etiskt försvarbart, måste personinformation hanteras och rutiner kring GDPR införas. Vid publicering till marknaden måste ytterligare undersökningar göras för att bestämma applikationens hållbarhet då aspekter som inte har analyserats i rapporten tillkommer.

Risker angående importeringen av externa paket måste även analyseras då kod kan ändras av andra användare vilket kan introducera en attackvektor framförallt i de paket som berör inmatning av information. När informationen sparas så kan i nuläget ett förändrat paket introducera skadlig kod i applikationen. Detta kan motverkas med robusta rutiner kring hantering av inmatad data samt utveckla alla inmatnings-widgets internt.

7 Slutsats

Utvärderingen av de implementerade användbarhetsökande lösningarna visar tydligt att implementationer som en rullbar digital klocka samt en delad meny, en förenklad meny som används tillsammans med en avancerad, medför en ökad användbarhet till funktioner som redan finns i kalenderapplikationer. Den visar även att nya funktioner som en algoritm som planerar åt användaren kan bidra med ökad användbarhet men om den ökar användbarheten beror på implementationen, där implementationer som en algoritm med viktning ökar användbarheten men sämre alternativ som en rent slumpad algoritm kan minska användbarheten.

8 Källförteckning

[1] Y. Huang, Z. Yang, and V. G. Morwitz, “How using a paper versus mobile calendar influences everyday planning and plan fulfillment,” *Journal of Consumer Psychology*, vol. 33, no. 1, pp. 115–122, May 2022. doi:10.1002/jcpy.1297

[2] E. Admin, “70% of adults rely on digital calendar,” ECAL, Accessed: May. 30, 2024. [Online]. Available: https://ecal.com/70-percent-of-adults-rely-on-digital-calendar/

[3] R.A.Fabio and T. Capri, “The executive functions in a sample of Italian adults with ADHD:attention, response inhibition and planning/organization,” in Mediterranean Journal of Clinical Psychology MJCP, vol 5 N 3, 2017. [Online]. Available: https://cab.unime.it/journals/index.php/MJCP/article/view/1636/pdf

[4] M. M. Sebrechts and J. B. Black, “Software psychology: Human factors in computer and information systems. Ben Shneiderman. Cambridge, Mass.: Winthrop, 1980. Pp. xiv + 320.,” Applied Psycholinguistics, vol. 3, no. 4, pp. 373–381, Dec. 1982, doi: 10.1017/s0142716400004331.

[5] J. Löwgren and E. Stolterman, “Design av informationsteknik : materialet utan egenskaper”. Lund: Studentlitteratur, 2004, pp 193-197.

[6] A. Cooper, R. Reimann, and D. Cronin, “About Face: The Essentials of Interaction Design”, 4th ed. Indianapolis, In: John Wiley And Sons, 2014, pp 140-156.

[7] R. Molich and J. Nielsen, “Improving a human-computer dialogue,” *Communications of the ACM*, vol. 33, no. 3, pp. 338–348, Mar. 1990, doi:10.1145/77481.77486.

[8] J. Nielsen, “10 Heuristics for User Interface Design,” Nielsen Norman Group, Accessed: August . 12, 2024. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/

[9] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI ’90, Mar. 1990, doi:10.1145/97243.97281.

[10] J. Nielsen “Usability Engineering”. Elsevier, 1993.

[11] S. Sharma, “Data Privacy And Gdpr Handbook”, John Wiley & Sons Incorporated, 2019.

[12] S.O. Hansson, "Teknik och etik," Kungl Tekniska Högskolan, Stockholm, Sweden, 2009. [Online]. Available: https://people.kth.se/~soh/tekniketik.pdf, Accessed: May. 25, 2024

[13] F. Hedenus, M. Persson, F. Sprei, “Hållbar utveckling”, Lund: Studentlitteratur, 2018.

[14] O. Ishimura and Y. Yoshimoto, "Just-In-Time Compiler System in Aspect-Oriented Programming based Building Block Platform for constructing Domain-Specific Language for HPC Application," 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW), Himeji, Japan, 2022, pp. 241-247, doi: 10.1109/CANDARW57323.2022.00082

[15] “The Dart Type System,” Dart. Accessed: May. 30, 2024. [Online]. Available: https://dart.dev/language/type-system

[16] P. Telangi, “How to create packages for flutter: A developer’s guide,” Mobisoft Infotech, Accessed: may. 30, 2024. [Online]. Available: https://mobisoftinfotech.com/resources/blog/how-to-create-packages-for-flutter-a-developers-guide/

[17] C. Chaubey and A. Sharma, “The Integrated Development Environment (IDE) for application development: Android Studio and its Tools,” *INSTRUMENTATION ENGINEERING, ELECTRONICS AND TELECOMMUNICATIONS – 2021 (IEET-2021): Proceedings of the VII International Forum*, 2023. doi:10.1063/5.0116494

[18] B. Johnson, “Visual studio code : end-to-end editing and debugging tools for web developers”, Indianapolis: Wrox, 2019.

[19] M. Tsitoara, “Beginning Git and GitHub. Berkeley”, CA: Apress, 2020. doi:10.1007/978-1-4842-5313-7.

[20] F. D’Urso, C. Santoro, and F. F. Santoro, “Wale: A solution to share libraries in Docker containers,” *Future Generation Computer Systems*, vol. 100, pp. 513–522, Nov. 2019. doi:10.1016/j.future.2019.03.049

9 Bilagor

9.1 Bilaga 1. Gantt schema

1. Skapa en app som kan visa ett schema för en dag.
2. Låta användare lägga till aktiviteter för den visade dagen med start- och slut-tid.
3. Implementera menysystem.
4. Låta appen visa flera dagar på veckoformat.
5. Låta användare lägga till aktiviteter för alla dagar, även de som inte visas med start-och slut-tid.
6. Implementera en grundläggande planeringsalgoritm.
7. Låta användare lägga till aktiviteter med endast längd och/eller intervall.
8. Implementation av månadsvy.
9. Implementera en vy för att kunna se vad som är hårt planerat av användaren och vad algoritmen har planerat.
10. Implementera en avancerad planeringsmeny med flerparametrar .
11. Låt användaren välja mellan dagsvy, månadsvy eller veckovy med olika gränssnitt. (F3)
12. Implementera visuell feedback samt notiser.
13. Implementera icke-visuell feedback.
14. Låt användaren välja mellan några tider som algoritmen valt.
15. Låta användare spara preferenser som används av algoritmen.
16. Möjliggöra inlärning för algoritm.
17. Implementera omplanering för aktiviteter.
18. Implementera förslag för schemaändringar om ingen tid hittas av algoritmen.
19. Implementera projektplaneringshantering för algoritm.
20. Delning av schema.
21. Grupp-planering för algoritm.

Uppgift\Vecka	8	9	10	11	12	13	14	15	16	17	18
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											

9.2 Bilaga 2. Resultat av undersökning

Build	Hur väl för	Hur väl mo	Hur begär	Hur lätt va	Hur mycket hjälp beh	Hur väl efti	Hur lätt va	Hur lätt va	Hur mycke	Om du gjo	Hur mycket hjälp beh	Hur lätt va	Hur väl stä	Hur lätt va	Hur ofta be	Hur mycke	Hur mång	Om du gjo	Hur mycke
FA	5	2	1	5	4	3	5	2	2	4	4	3	5	2	5	2	5	5	5
FA	5	2	3	5	4	2	5	1	3	5	4	4	5	5	5	2	4	5	5
FA	5	1	1	5	5	1	2	1	4	4	3	1	5	5	4	4	5	4	3
FA	5	1	2	5	5	4	5	4	3	5	4	3	5	5	5	2	5	5	5
FA	5	1	2	5	4	3	5	2	3	5	4	3	5	4	4	2	5	5	5
AVG	5	1,4	1,8	5	4,4	2,6	4,4	2	3	4,6	3,8	2,8	5	4,2	4,6	2,4	4,8	4,8	4,6
KEI	2	4	4	4	5	4	3	4	1	5	5	3	5	4	4	2	5	4	4
KEI	5	3	1	4	2	4	3	3	4	3	5	3	5	4	4	3	4	3	4
KEI	2	4	2	5	4	4	5	4	4	4	5	1	2	2	2	4	4	3	2
KEI	4	4	3	5	5	5	4	4	3	4	5	2	5	3	4	3	5	3	3
KEI	3	3	3	5	4	4	5	4	3	4	5	2	4	3	4	3	5	3	3
AVG	3,2	3,6	2,6	4,6	4	4,2	4	3,8	3	4	5	2,2	4,2	3,2	3,6	3	4,6	3,2	3,2
ST	5	3	5	5	5	1	5	4	4	5	5								
ST	5	1	3	4	5	1	4	5	2	5	5								
ST	2	1	1	5	2	4	5	2	3	3	4								
ST	3	1	2	5	5	2	4	4	3	5	5								
ST	3	1	2	4	4	2	5	4	3	5	5								
AVG	3,6	1,4	2,6	4,6	4,2	2	4,6	3,8	3	4,6	4,8								
KMI	1	4	4	4	4														
KMI	5	2	1	4	2														
KMI	3	4	2	5	4														
KMI	4	4	3	5	5														
KMI	3	4	3	4	4														
AVG	3,2	3,6	2,6	4,4	3,8														