

**JÄMFÖRELSE MELLAN JOOMLA OCH
REACT.JS UTIFRÅN SVARSTID PÅ EN
BOKHANDEL WEBBSIDA**

**COMPARISON BETWEEN JOOMLA AND
REACT.JS BASED ON RESPONSE TIME ON
A BOOKSTORE WEBSITE**

Examensarbete inom huvudområdet
Informationsteknologi
Grundnivå 30 Högskolepoäng
Vårtermin 2021

Abukar Ahmed (a18abuah)

Handledare: Henrik Gustavsson
Examinator: Mikael Berndtsson

Sammanfattning

I denna studie jämförs JavaScript-biblioteket React mot Joomla på en e-commerce webbsida. Tillämpningen på e-commerce webbsidan är bokhandel. I implementationen skapades det två webbsidor. En med React och den andra med Joomla. Webbsidan använder sig även av Bootstrap som CSS-ramverk för att få samma layout. Detta är viktigt då mätningen utförs i en kontrollerad miljö där det är viktigt att allting utgår ifrån samma förutsättningar. Mätningen sker genom ett eget skript som körs genom Tampermonkey tillägget i Chrome. Resultatet av mätningarna visas senare i grafer som indikerar att React har en mycket snabbare svarstid jämfört med Joomla-webbsidan. I framtida arbeten går det att öka antalet produkter och innehåll och göra webbsidan mer verklighetsbaserat. Det går även att byta tillämpningstema till en elektronikwebbsida där det förmodligen skulle behöva innehålla fler bilder och även videoklipp.

Nyckelord:[React, Joomla, Bootstrap, E-commerce, Bokhandel]

Innehållsförteckning

1. Introduktion	1
2. Bakgrund	2
2.1 CMS	2
2.1.1 Joomla	2
2.2 E-commerce	3
2.3 Ramverk och bibliotek	4
2.3.1 Bootstrap	4
2.3.2 React	5
2.4 Tillämpning	6
3. Problemformulering	8
3.1 Frågeställning	8
4. Metod	10
4.1 Etik	11
5. Genomförande	13
5.1 Litteraturstudie	13
5.2 Installation	14
5.3 Progression	15
5.3.1 Joomla template	16
5.3.2 Joomla module	18
5.3.3 React render	21
5.3.4 React JSON	23
5.3.5 Automatisering skript	26
5.3.6 Svarstid-mättnings skript	28
5.3.6.1 React	28
5.3.6.2 Joomla	30
6. Pilotstudie	32
6.1 Diskussion av pilotstudie	34
7. Utvärdering	35
7.1 Presentation av testfall	35

7.2 Analys av testfall	36
7.2.1 Testfall 1	36
7.2.2 Testfall 2	37
7.2.3 Testfall 3	39
7.2.4 Testfall 4	40
8. Avslutande diskussion	42
8.1 Sammanfattning	42
8.2 Diskussion	43
8.3 Etik och samhälle	44
8.4 Framtida arbete	45
Referenser	47

1. Introduktion

E-commerce eller e-handel är när ett företag eller en privatperson handlar varor via internet. Under det senaste decenniet har e-commerce ökat kraftigt. I och med ökningen av populariteten har även standarden för hur en e-commerce ska se ut ökat. Enligt forskningsartikeln skriven av Mickens (2010) förväntar sig användaren att sidan ska ladda i max 2 sekunder. Detta blir då ett problem om webbsidan överstiger dessa svarstider. Om webbsidans svarstider överstiger 2 sekunder kan användaren tappa förtroende för webbsidan och därmed byta till en annan sida som fungerar bättre.

Det finns många olika tekniker för att skapa en e-commerce. En av teknikerna är CMS. CMS står för content management system och är en enkel teknik att skapa hemsidor på enligt Mirdha et al. (2014). Enligt Mirdha et al. (2014) är CMS en mjukvara som hjälper människor utan programmeringskunskaper att skapa, uppdatera och publicera innehåll på en webbsida. Det finns många olika CMS att välja mellan som besitter olika fördelar och nackdelar. I detta arbete skapas en webbsida i Joomla. Joomla är det bästa CMS:et på interaktiva och stora hemsidor enligt både Patel et al. (2011) och Mirdha et al. (2014). En annan teknik inom skapandet av e-commerce är JavaScript-ramverk. JavaScript-ramverk skapades ihop med Chrome v8 engine som gjorde det möjligt för JavaScript att ändra mer än CSS på webbsidor enligt Xing et al. (2014). CSS står för Cascading Style Sheets och det används för att byta, typsnitt, textstorlek, färg och layout på webbsidan. I detta arbete används React som JavaScript-ramverk. Ramverket React har en hög prestanda när det kommer till större webbsidor.

I studien skriven av Kiatruangkrai et al. (2010) skapas det en e-commerce med hjälp av ett CMS. Enligt författaren används CMS av företag som inte har tillgång eller resurser till programmerare. Med CMS behövs det inte programmerare för att skapa och uppdatera en CMS-webbsida. Problemet med CMS kan därmed vara långsamma svarstider enligt Tomisa et al. (2019). Detta förekommer på grund av alla filer som CMS laddar in. Frågan är om svarstiden skulle förbättras om företaget bytte teknik till React istället för Joomla om de har tillgång till programmerare.

Genomförandet sker genom skapandet av två webbsidor, en skapas med React och den andra skapas med Joomla. Experimentet sker i en kontrollerad miljö. En kontrollerad miljö bidrar till ett mer rättvist experiment. Det är därför båda webbsidorna som skapas måste se identiska ut. På detta vis blir det inte CSS som gör skillnaden i svarstiden. Båda webbsidorna ska använda Bootstrap som är ett CSS-ramverk med fördefinierade klasser.

Mätningen utförs med hjälp av ett skript som körs genom Tampermonkey (Tampermonkey.net 2021). Tampermonkey är ett tillägg i Chrome som kör skriptet direkt på en specifik webbsida. Skriptet som körs itererar genom produkterna och räknar ut svarstiden för varje navigation.

Innan genomförandet och mätningen av svarstiden utfördes det en litteraturstudie för att få fram information om båda teknikerna som ska vara till nytta under studien. I detta arbete användes det vetenskapliga artiklar, böcker och även guider. Hela arbetet är dokumenterat i denna rapport och all kod i genomförandet ligger uppe på GitHub för att göra replikering av studien vid framtida arbeten möjligt.

2. Bakgrund

2.1 CMS

CMS står för content management system. CMS beskrivs av Mirdha et al.(2014) som en mjukvara som hjälper personer utan kunskaper inom webbutveckling att skapa och hålla reda på innehållet i en webbsida. Med hjälp av CMS kan användaren byta funktionalitet på ett snabbt och simpelt sätt. Enligt Mirdha et al.(2014) skiljer sig detta från programmerad webbsida då den programmerade webbsidan är svårare att byta funktionalitet på än på ett CMS. Detta är en av de anledningar till varför stora företag och även statliga hemsidor som exempelvis The White House i usa använder sig av CMS.

Det finns olika CMS för användaren att välja emellan som exempelvis Joomla, drupal eller wordpress. Enligt Patel et al.(2013) beskriver författarna att alla CMS rent generellt har tre uppgifter att utföra. Den första uppgiften är att skapa webbsidan, den andra är att redigera webbsidan och det tredje är att publicera webbsidan .

2.1.1 Joomla

Joomla är ett CMS som är både enkelt att använda sig av och flexibelt enligt Mirdha et al.(2014). Enligt Cao et al.(2010) är Joomla en av de mest populära CMS:en. Med Joomla går det att skapa interaktiva hemsidor på ett snabbt sätt. Detta CMS har en stor gemenskap då många använder sig av Joomla. Denna gemenskap hjälps åt att lösa problem och vidareutveckla CMS:et. Joomla har vunnit flera priser sedan det startades upp år 2005. Enligt både Patel et al.(2011) och Mirdha et al.(2014) så är Joomla det mest flexibla CMS:et och det CMS:et som är bäst anpassat för interaktivitet och stora mängder av innehåll.

Joomla standard installation innehåller redan multifunktionella system men det går även att ladda hem mer. Enligt Cao et al.(2010) finns det 5 tillägg i Joomla. De 5 tilläggen är komponenter, modul, plugins, templates och språk.

- Komponent är den största och mest komplicerade tillägget. Varje komponent funktion har två delar, den administrativa och webbsidans. Ett exempel på en komponent kan vara en funktion som håller koll på inloggningarna på webbsidan. Komponenterna kan triggas av ett menyval.
- Modul är ett mer flexibelt och enkelt tillägg som används för att framställa en sida. Moduler kan vara länkade till komponenter som nämndes tidigare men de kan även bara bestå av vanlig html kod eller text.
- Plugins är tillägg som är mer avancerat tillägg. Plugins körs igång när en applikation den är kopplad till körs.
- Template är ett tillägg som bestämmer hur webbsidan ska se ut. Det finns flera olika templates att välja emellan. Det finns två typer av templates, frontend och backend. Det går även att redigera på hur template komponenterna ska placeras.
- Språk är ett tillägg som översätter statisk text strings som ligger i Joomla källkod. Tillägget består av antingen kärn paket eller Förlängnings paket. Språk tillägget påverkar både front och administrationssidan.

I Figur 1 visas det hur en webbsida redigeras och skapas i Joomla.

Menus: Edit Item

Save Save & Close Save & New Save as Copy Close

Joomla! would like your permission to collect some basic statistics.

To better understand our install base and end user environments it is helpful if you send some site information back to a Joomla! controlled central server. No idea that will be sent.

Enable Joomla Statistics?

Always Once Never

Menu Title * Home Alias homepage

Details Options Link Type Page Display Metadata Module Assignment

Menu Item Type * Single Article Select

Select Article * Getting Started Edit Clear

Link index.php?option=com_content&view=article&id=1

Target Window Parent

Template Style - Use Default -

Figur 1 Joomla menyval

2.2 E-commerce

E-commerce eller e-handel är när ett företag eller en konsument som säljer, köper eller byter en produkt eller tjänst via internet. E-commerce har blivit stort och fortsätter fortfarande växa med åren. På grund av detta blir det mycket konkurrens då de flesta företag använder sig av e-handel. Enligt Lanford et al.(2004) behöver e-commerce hemsidor flera olika kriterier för att några kunder ska kunna lita på webbsidan och komma tillbaka till den i framtiden. En av de kriterierna är just svarstiden som är hur snabbt det går för webbsidan att ladda information.

På grund av populariteten som har växt så växer även förväntningarna på hur en e-commerce webbsida ska fungera. Enligt Lanford et al.(2004) förväntar sig användare att innehållet på webbsidan ska ladda snabbt. Enligt Mickens (2010) förväntar sig användaren att sidan ska ladda i max 2 sekunder. Det visar sig även att 40% av användare inte väntar mer än 3 sekunder innan de lämnar webbsidan.

Det finns många tekniker för att skapa en e-commerce webbsida. De flesta teknikerna har behov av att någon har programmeringskunskaper. Enligt Mirdha et al.(2014) behöver inte personer som använder sig av CMS inneha några kunskaper inom programmering. Enligt Kiatruangkrai et al.(2010) är det många företag som börjar använda sig mer och mer av CMS när det kommer till skapandet av e-commerce hemsidor. Detta kan bero på att företagen inte har tillgång till programmerare eller resurser för programmerare. Enligt Mirdha et al. (2014) är det även enklare att ändra detaljer om man använder CMS jämfört med en programmerad

webbsidan. Ett exempel när det kommer till e-commerce kan vara om webbsidan har rea på vissa varor eller nya varor ska läggas till. Detta blir mycket enklare att hantera med hjälp av ett CMS.

2.3 Ramverk och bibliotek

Ramverk och bibliotek är till för att underlätta för utvecklare att skapa webbsidor och applikationer. Enligt Xing et al.(2019) finns det många ramverk och bibliotek för utvecklare att välja mellan när det kommer till e-commerce. Valet handlar om att maximera användarupplevelsen. Enligt Xing et al.(2019) framkom ramverk baserade på JavaScript efter att google utvecklats Chrome v8 engine. Fram tills dess var JavaScript mer för att byta CSS på webbsidan. Med hjälp av v8 engine gick det att göra mer med JavaScript än att bara byta CSS.

2.3.1 Bootstrap

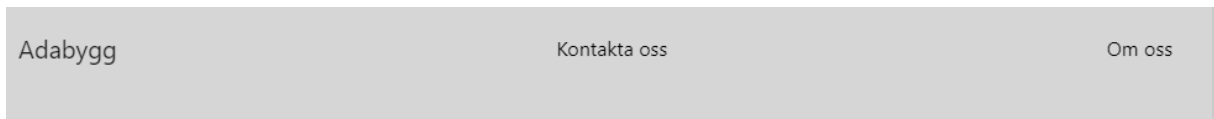
Enligt Jiang et al.(2014) är Bootstrap ett open source front-end ramverk som var utvecklades av Twitter. Bootstrap är ett CSS/HTML ramverk till skillnad ifrån JavaScript ramverk som React. Enligt Jiang et al.(2014) erbjuder Bootstrap eleganta specifikationer i HTML och CSS med färre rader kod. Elementen som skrivs är redan definierade för användare att bara kalla på. Bootstrap är ett JavaScript plugin som bara är 1.7 kb vid användning. Bootstrap stödjer CSS3 transitioner från "height: 0" till "height: auto" som inte finns med i CSS3 transitioner. Enligt Intal et al.(2018) är Bootstrap ett "mobile first" ramverk som är till för att göra webbutveckling snabbare. Det är "mobile first" då ramverket automatiskt anpassar sig till allt från skärm, surfplattor och mobiler. Detta blir viktigt i studien som Jiang et al.(2014) utför där en responsiv e-commerce webbsida ska skapas. Med Bootstrap blir CSS i webbsidan responsiv utan att sätta några begränsningar till storlek på innehåll i webbsidan. Enligt Jiang et al.(2014) har responsiv design på webbsidan blivit viktigt under de senaste åren. Responsiv design innebär att användare med olika relationer ska kunna se samma innehåll på webbsidan. När detta inte går igenom blir det svårt för användare med mindre upplösning att bläddra i stora e-commerce hemsidor. I Figur 2 går det att se hur koden till en Bootstrap navbar ser ut.

Först måste användaren länka till Bootstrap för att kunna använda den fördefinierade HTML och CSS koden. Efter detta skapas det en vanlig navbar. I navbaren finns det många fördefinierade deklARATIONER för hur navbaren ska se ut.

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">Adabygg</a>
    </div>
```

Figur 2 Bootstrap navbar

I Figur 3 och 4 visas den output som är definierad av programkoden i Figur 2. I Figur 3 är webbsidan tillräckligt stor i bredden för att visa alternativen på navbar, men i Figur 4 är webbsidan inte tillräckligt bred. Bootstrap gömmer då navbaren och visar det objektet som syns i figur 4 som går att klicka på för att visa resten av alternativen i navbaren.



Figur 3 Navbar i Bootstrap med hela längden



Figur 4 Navbar i Bootstrap för skärmar med mindre upplösning

2.3.2 React

React är ett JavaScript bibliotek som är till för att skapa kombinerbara användargränssnitt enligt Intal et al.(2018). JavaScript biblioteket uppmuntrar utvecklare att använda sig av återanvändbara UI komponenter som visar data som ändras över tid. Enligt Xing et al.(2014) skapades React av facebook för att öka användarupplevelsen i facebook och instagram. Eftersom React var en succé släppte även facebook React som en open source JavaScript bibliotek för utvecklare och verksamheter år 2013.

React har en speciell teknologi för rendering som gör webbsidan snabbare enligt Xing et al.(2014). Den gör först om webbsidans innehåll till olika komponenter i en DOM (document object model). När webbläsaren kör webbsidan kommer den nu köra komponenterna med JavaScript. Att köra komponenterna med JavaScript är snabbare än den traditionella dynamiska webbsidan. En annan funktion i React biblioteket är virtuella DOM. Den virtuella DOM höjer svarstiden när användaren uppdaterar webbsidan eller bläddra till en sub-sida. Detta görs med hjälp av one-way data binding. När användaren bläddra till en annan sub-sida eller uppdaterar webbsidan skapar React en virtuell DOM av den webbsidan användaren befann sig innan hen klickade på den andra webbsidan. När webbläsaren ska visa den nya webbsidan jämförs den virtuella DOM med sub-sidans DOM. Om det finns likadana komponenter behöver dessa inte bli uppdaterade, istället hämtar den DOM komponenter som inte existerade på förra sidan. Detta höjer svarstiden hos React applikationer och hemsidor.


Olika front-end ramverk och bibliotek har olika fördelar och svagheter. React fördelar när det kommer till e-commerce är den höga effektiviteten när det kommer till rendering av webbsidan. Denna rendering gör webbläsaren snabbare genom att köra den igenom JavaScript istället för den traditionella renderingen som är långsammare. Detta gör det lättare för utvecklare att inte bekymra sig om att uppdatera varorna eller lägga till varor. En annan fördel med React när det kommer till e-commerce är att det går att utveckla

applikationer för webbsidan med hjälp av React native mobile efter att utvecklaren har lärt van med att programmera i React.

I Figur 4 är det React kod som skriver ut hello world på en webbsida. Först skrivs det ut en div som är tom. Efter det fylls denna tomma div med hjälp av ReactDOM som skriver in "hello world" med hjälp av render funktion. i Figur 6 visas detta outputen som skrivs ut efter att koden i Figur 5 har blivit utförd.

```
ReactDOM.render(  
  <h1>Hello, world!</h1>,  
  document.getElementById('root')  
)
```

Figur 5 React kod.



Figur 6 Output från React koden i Figur 5.

2.4 Tillämpning

E-commerce blir ett alltmer effektivt sätt att sälja varor på. För att skapa en e-commerce behövs det programmeringskunskaper enligt Kiatruangkrai et al.(2010). I de små företagen finns det inte lika många anställda och resurser. Enligt Kiatruangkrai et al.(2010) är detta anledningen till varför många väljer att skapa e-commerce hemsidor med hjälp av CMS. I Kiatruangkrai et al.(2010) studie skapas det ett CMS som är till för att skapa en e-commerce webbsida för personer utan programmeringskunskaper. Efter det utförs det ett experiment där personer själva ska skapa en e-commerce webbsida med hjälp av det nya CMS:et. Men enligt Voutilainen et al.(2015) går det att skapa en CMS sida med hjälp av ramverket Bootstrap. I studien bygger författaren en informationssida som lagrar olika typer av information. webbsidan använder sig av Bootstrap för responsiviteten som kommer med ramverket. Detta gör webbsidan mycket mer flexibel när det kommer till olika skärmar som mobiler eller surfplattor.

En brist från studien Kiatruangkrai et al.(2010) är att den inte tydliggör vilken slags e-commerce webbsidan författaren bygger. Det finns många olika typer av e-commerce hemsidor och för varje typ behövs det olika mycket information och bilder. Det är därför denna studie utgår ifrån e-commerce typen bokhandel som utförs i artikeln skriven av Liang-fu et al.(2010). I artikeln skapar Liang-fu et al.(2010) en e-commerce för en bokhandel.

I e-commerce webbsidan lagras det information för varje respektive bok. För varje bok visas det information om boken och även ett kommentarsfält som både oregistrerade och registrerade användare kan kommentera på. Informationen som lagras på webbsidan visas på Figur 7.

Field name	Field Type	If empty
BookID	int	NOT NULL
CategoryID	int	NOT NULL
BookName	nvarchar(100)	NULL
Author	nvarchar(200)	NULL
Translator	nvarchar(50)	NULL
UnitPrice	money	NULL
Publisher	nvarchar(50)	NULL
BindAndLayout	nchar(10)	NULL
Pages	int	NULL
PubDate	datetime	NULL
Language	nvarchar(20)	NULL
ISBN	nvarchar(50)	NULL
BookDescription	nvarchar(max)	NULL
AuthorDescription	nvarchar(max)	NULL
Catalog	nvarchar(max)	NULL
Forewords	nvarchar(max)	NULL
TotalRating	int	NULL
AddDate	datetime	NULL
AddBy	nvarchar(50)	NULL
Quality	int	NULL
BookImageURL	nvarchar(500)	NOT NULL
ShowToIndex	bit	NULL
DiscountPrice	money	NULL
ViewCount	int	NULL

Figur 7 Informationen som sparas om varje bok

3. Problemformulering

I studien Kiatruangkrai et al.(2010) skapas det en e-commerce webbsida med hjälp av CMS. Författaren förklarar att CMS är till för verksamheter utan resurser för programmerare. I en annan studie visar det sig även att CMS kan användas ihop med Bootstrap för att förbättra responsiviteten på webbsidan (Voutilainen et al.2015). Ett problem som kan förekomma med användandet i CMS är långa svarstider enligt Tomisa et al.(2019). De långa svarstiderna förekommer på grund av att webbsidan måste ladda in många filer som exempelvis tema filer. När användaren öppnar webbsidan får hen inte resultatet förrän servern har skickat iväg alla filer som kan vara många beroende på vilket CMS som används och hur många filer som laddar. Även fast CMS erbjuder en tjänst där det går att skapa webbsidor på ett enkelt och smidigt sätt kan detta bli betydelselöst ifall webbsidan ger en dålig svarstid som det beskrivs i studien Tomisa et al.(2019).

Frågan är då om personen i fråga har tillgång till programmerare och resurser, skulle det vara värt att byta teknik från CMS till React för att ge en bättre svarstid. Om webbsidan som skapades i Kiatruangkrai et al.(2010) istället skapades med React, skulle detta kunna resultera i bättre svarstider? Denna fråga är viktigt då svarstid är en mycket viktig faktor. I Mickens (2010) står det hur viktigt det är för kunderna att få svar från webbsidan under några sekunder. Om sidan är seg kan kunderna förlora förtroende för sidan och köpa ifrån en annan sida istället. Det är därför det är viktigt att jämföra CMS:et Joomla med React för att se om svarstiden skiljer sig och om det är värd att byta teknik på sin e-commerce webbsida.

Syftet med denna studie är att undersöka ifall React kommer att förbättra svarstiden hos en e-commerce webbsida jämfört med Joomla. Då det finns en stor risk för att förlora kunder om webbsidan presterar sämre och att standarden för e-commerce hemsidor ökar blir detta en viktig fråga att svara på. Hemsidorna ska även se identiska ut för att det inte ska vara layouten som ger bättre svarstiden. Detta görs med hjälp av Bootstrap som studien skriven av Voutilainen et al.(2015) visar. Enligt Jiang et al.(2014) visar det sig att ramverket Bootstrap bara tar 1.7 kb på webbsidan som använder ramverket. Detta blir då jämligt mellan båda hemsidorna som ska skapas under denna studie.

Anledningen till varför React jämförs mot Joomla är för att teknikerna inte har blivit jämförda i någon tidigare studie utifrån en e-commerce webbsida. Joomla har blivit jämfört med andra CMS Mirdha et al.(2014) och React har blivit jämfört med andra JavaScript ramverk. Båda alternativen visade sig vara bäst anpassade för en e-commerce webbsida utifrån sina tekniker. Det är därför det är viktigt att jämföra båda alternativen för att se vilken av de som är bäst anpassat för just den e-commerce webbsidan som ska byggas i denna studie.

3.1 Frågeställning

Denna studie ska svara på frågeställningen om React ger bättre svarstid än CMS:et Joomla på en bokhandel e-commerce webbsida. Hypotesen är att React kommer att förbättra svarstiden hos e-commerce webbsidan .

Hypotesen stöds av Nikulchev et al.(2018) där författaren genomförde ett experiment för olika ramverk. Där kom författaren fram till att React ramverket har en bra prestanda jämfört med andra ramverk. Anledningen till varför Joomla förmodligen inte kommer att prestera lika bra på en webbsida är för att den sparar ner mindre cache enligt Patel et al.(2013). Caching gör webbsidan snabbare med att kopiera data på webbläsaren. Den data kan antingen vara data som användaren ofta använder sig av eller data som användaren ofta använder sig av. Nästa gång användaren kommer in på webbsidan kommer webbsidan snabbt kunna visa upp det som ligger i cache minnet. Denna teknik används i webben för att förminska serverfördröjningen. En annan anledning är att CMS laddar hem många filer som fördröjer svarstiden hos användaren enligt Tomisa et al.(2019).

4. Metod

Metoden som kommer att användas i detta examensarbete kommer vara ett tekniskt orienterat experiment. Anledningen till varför detta ska vara ett experiment är för att ta reda ifall Joomla eller React är mer anpassade för skapandet av e-commerce webbsida.

Detta experiment kommer utföras i en kontrollerad miljö. Enligt Wohlin et al.(2012) bidra kontrollerade miljöer till lika förutsättningar. Båda experimenten ska utgå ifrån samma specifikationer för att resultatet ska bli så rättvist som möjligt. Ett exempel är om det ena experimentet sker på en viss dator med ett visst bredband. Då ska samma experimentet ske med samma förutsättningar för att resultatet ska vara rättvist.

I detta examensarbete utförs det med metoden experiment. En alternativ metod som kunde använts istället för experiment är undersökning. Wohlin et al.(2012) Förklarar att metoden undersökning är mer baserat på användare. Användare ska exempelvis testa på ett system och sedan svara på frågor. Resultatet från undersökningen undersöks vidare för att härleda till en beskrivande sammanfattning till varför resultatet blev som det blev. Anledningen till varför denna metod inte kommer användas är för att svarstiden som hela experimentet är fokuserat kring är på millisekunder. En människa kommer inte kunna anmärka exakt hur mycket snabbare en webbsida är med millisekunder som scriptet gör. En annan grej är också att i detta examensarbete handlar det mer om svarstiden hos ramverket eller CMS:et istället för användarvänligheten. Om detta var tanken skulle experimentmetoden vara bättre tillämpad.

En annan alternativ metod som kunde utförts i på detta experiment är fallstudie. Enligt Wohlin et al.(2012) är en fallstudie en observationsstudie som är motsatsen till en kontrollerad studie som är experimentet. Anledningen till varför denna metod valdes bort i denna examensarbete var på grund av att med en kontrollerad miljö blir datan mycket enklare att dra slutsatser ifrån. Om en fallstudie används skulle det inte gå veta om det var på grund av datorn, uppkopplingen eller nätverket som var problemet eller om själva webbsidan var segare. Detta blir viktigare i denna examensarbete på grund av att det kan skilja endast millisekunder mellan hemsidorna. Då är det viktigt att båda hemsidorna har samma förutsättningar.

I detta experiment valdes det att mäta på mjukvaran vilken webbsida som är snabbast. Detta görs med ett skript som kommer att räkna tiden som det tar för webbsidan att ta sig ifrån en sida till en annan. Med mätningen kan det då dras slutsatser om vilken av de två ändringarna som har snabbast svarstid.

En alternativ undersökningsmetod som inte används i detta examensarbete är enkätundersökningar. Enkätundersökningar är en undersökningsmetod där användaren får svara på enkäter. Enligt Wohlin et al.(2012) brukar enkätundersökningar handla mestadels om när en process har blivit förbättrad eller ändrad. Användaren måste då redan använt webbsidan innan förbättringen. Denna undersökningsmetod valdes bort på grund av att den inte är kontrollerad nog för att utföra. Om användare över hela världen gör test mot webbsidan så kommer det inte gå att veta vad dessa resultat beror på. Det kan vara något

problem med hårdvaran eller mjukvaran hos användaren och därmed blir det svårt att sammanställa och beskriva varför resultatet blev som det blev. Enkätundersökningen skulle passa mer om examensarbetet handlade om webbsidans användarvänlighet.

En annan undersökningsmetod är genom intervju. Intervjumetoden är lik enkätmetoden. Skillnaden enligt Wohlin et al.(2012) är att med intervju blir det generellt mindre “jag vet inte” eller “nej” svar då intervjuaren kan svara på frågor om en fråga skulle vara oklar. Intervjuundersökningar får mer svar än enkätundersökningar enligt Wohlin et al.(2012). Anledningen till varför denna undersökningsmetod inte blev vald är för att det inte heller sker i en kontrollerad miljö. Utan den kontrollerade miljön blir sammanställningen av svar svårare. Intervju blir bättre anpassad med en examensarbete som handlar mer om användarvänlighet än svarstid från en webbsida.

I denna studie ska ett experiment utföras med hjälp av ett skript som mäter svarstid. skriptet ska först bläddra till en annan sida och sedan mäta tiden det tog för den att ladda. Tiden kommer att sparas i millisekunder i local storage och då kan grafer bli skapade av dessa. Då experimentet ska utföras i en fixed miljö så kommer de enda faktorerna vara React och Joomla. Enligt Wohlin et al.(2012) beror detta på att i ett fast experiment miljö ska bara en variabel bli manipulerad och i detta fall är det tekniken som används för att skapa webbsidan. Den variabeln kallas även för faktor. De andra variablerna som nätverksuppkoppling eller ram-minne på datorn som experimentet utförs på ska inte manipuleras. Med detta får vi ut anledningen till svarstiden då detta bara kan bero på faktorerna som i detta fall är React och Joomla.

En alternativ faktor kan vara att sätta in olika mängd produkter på webbsidan. Med denna faktor behövs det fortfarande faktorn där teknik byts. Faktorn kommer att ge svaret om vilken webbsida som är bäst anpassad för vilken slags verksamhet. Om exempelvis en teknik klarar sig bättre med liten data men blir segare med mer data betyder detta att den är mer anpassad för en mindre verksamhet. En annan alternativ faktor kan vara att variera på informationen för produkterna. Enligt Liang-fu et al.(2010) studie visas det hur mycket information som matas in för varje bok. Några exempel av dessa är bild, kvalitet och rating. Denna information kanske kan tas bort för att testa med olika mängder data för att se vilken av de två ramverken som är bäst anpassad för mindre data och vilken av ramverken som fortfarande ger bra svarstid oavsett datamängd.

4.1 Etik

För att detta projekt ska kunna replikeras kommer den fullständiga koden vara tillgänglig på GitHub och i appendix. Alla servrar och databaser kommer att vara skapade under experimentets gång. Detta experiment kommer inte att vara människoberoende och därmed behövs mänskliga faktorer räknas in i etiken.

Problem som kan uppstå under experimentets gång är om data som skrivs in på webbsidan är upphovsskyddat. Detta kan leda till rättsliga problem ifall företaget som har upphovsrätt upptäcker att det används utan tillåtelse. För att undvika dessa problem ska data som är upphovsskyddat ska undvikas.

För att säkerställa arbetets trovärdighet kommer experimentet utföras under samma förutsättningar. Detta är viktigt då svarstiden kan variera beroende på hårdvara och mjukvara. Det kan även variera under olika nätverksuppkopplingar. I denna studie kommer det bara att utföras på operativsystemet windows och i webbläsaren Chrome. Detta är för att miljöerna som detta arbete kan utföras på är för många. Det kan förekomma att någon miljö ger olika svarstider. Ett exempel kan vara om mac skulle ge sämre svarstider på React än windows. De olika miljöerna kan testas i framtida arbeten.

I denna studie kommer datan vara genererad. Detta kan leda till att experimentet blir mindre verkligt. Problemet är att om riktigt data skulle användas kan det leda till brott mot upphovsskyddat material.

5. Genomförande

5.1 Litteraturstudie

För att underlätta implementering i detta arbete togs det inspiration och hjälp från många artiklar och böcker för att verkställa resultatet i både Joomla och React.

I boken skriven av Wohlin et al.(2012) beskriver författarna hur experiment utförs i software engineering. Boken presenterar olika metoder för att utföra experiment som bland annat experiment i kontrollerad miljö som används i detta arbete. I boken presenteras det även olika utvärderingsmetoder för att få in data från experimentet. I detta arbete används det av utvärderingsmetoden mätning på mjukvaran som också var presenterat i boken.

I forskningsartikeln skriven av Cao et al.(2010) går författarna igenom de 5 tilläggen som Joomla erbjuder. I denna forskningsartikel ligger fokuset på att presentera vad de olika tilläggen har för funktion. I artikeln förekommer det inte en guide som visar hur de olika tilläggen kan tillämpas på en webbsida. För att uppnå detta används Joomla! Documentation (2016) och Joomla! Documentation (2019). I Joomla! Documentation (2016) förklarar författaren hur användare kan skapa en egen template. Artikeln är en steg för steg guide som förklarar och även visar upp hur det slutliga resultatet ser ut. Det är även ifrån den artikeln som gav inspirationen att skapa ett egen template istället för att använda färdiggjorda. I Joomla! Documentation(2019) förklarar författaren vad en module är och hur det går att placera den i en template. Även denna artikel är en steg för steg guide som förklarar implementationen med bilder för att visa hur det ska se ut. I artikeln förekommer det även de olika förinstallerade modulerna som redan finns i Joomla projektet och en förklaring på deras funktion.

I artikeln skriven av Lewis (2017) beskriver författaren hur React kan bli använt utan npm Babel eller en webbpaket. I artikeln beskriver författaren fördelarna med att inte använda sig utav alla de olika verktygen och ramverken runt om React. I React förekommer det även ett kodexempel på hur React kan användas utan alla de verktygen som sedan går att replikera och utgå ifrån. Det var även ifrån denna artikel inspirationen av att köra React utan tillägg eller ramverk kom. Med hjälp av användandet av React utan ramverk och andra verktyg blir det enklare att jämföra med Joomla.

I boken skriven av Gackenhimer (2015) presenteras det vad React för läsaren och även hur det går att implementera de olika delarna av ett React projekt. Första kapitlet handlar om vad som gör React till ett populärt JavaScript ramverk. Därefter presenteras det hur de olika delarna av React ramverket bidrar till att det är populärt. I boken presenteras det även några inbyggda funktioner som exempelvis `componentDidMount()` som användes i detta arbete.

För att uppnå en dynamisk webbsida krävdes det att få in produkter med JSON. Detta beskrivs tydligt i artikeln Sharma (2020). I Sharma (2020) går författaren igenom hur JSON data kan laddas in i React komponenter. I exemplet tar författaren in stock datafil i en JSON fil som sedan laddas in i en webbsida med hjälp av React. Artikeln går igenom steg för steg vilken kodrad som gör vad och hur det hela ska se ut i implementationen.

När implementationen genomfördes förekom det problem med artiklarna och böckerna som användes för att skapa React webbsidan. Då React i vanliga fall används ihop med node var det viktigt med dokumentationen från React webbsidan (Reactjs.org 2020) som visade kodsntuttar som hjälpte implementationen. Det var även med hjälp av (Reactjs.org 2020) dokumentationen som nya användbara funktioner implementeras som exempelvis `componentdidmount()` som är en funktion som blir tillkallad efter renderingen av webbsidan. Den funktionen användes vid svarstiden testningen av React webbsidan.


I implementationen av båda hemsidorna förekom det även många fel som inte hade någon förklaring i litteraturen eller i dokumentationen på hemsidorna. Det var då webbplatsen stackoverflow användes. Stackoverflow är en fråga och svar webbsida där många kunniga programmerare svarar på frågor angående programmering. Ett exempel är (stackoverflow 2021) där frågan handlade om hur rendering av navbar ser ut på en React webbsida.

5.2 Installation

För att kunna implementera Joomla webbsidan krävs det att program laddas hem. Först ska XAMPP laddas ner. XAMPP installeras ifrån (D 2021). Då implementationen utfördes på en windows 10 laddas XAMPP version 8.0.3 för windows.

Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

 **XAMPP for Windows 7.3.27, 7.4.16 & 8.0.3**

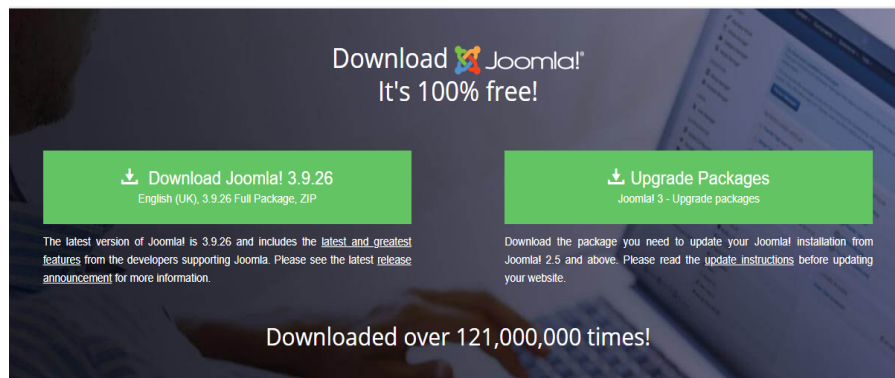
Version		Checksum		Size
7.3.27 / PHP 7.3.27	What's Included?	md5 sha1	Download (64 bit)	154 Mb
7.4.16 / PHP 7.4.16	What's Included?	md5 sha1	Download (64 bit)	156 Mb
8.0.3 / PHP 8.0.3	What's Included?	md5 sha1	Download (64 bit)	156 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

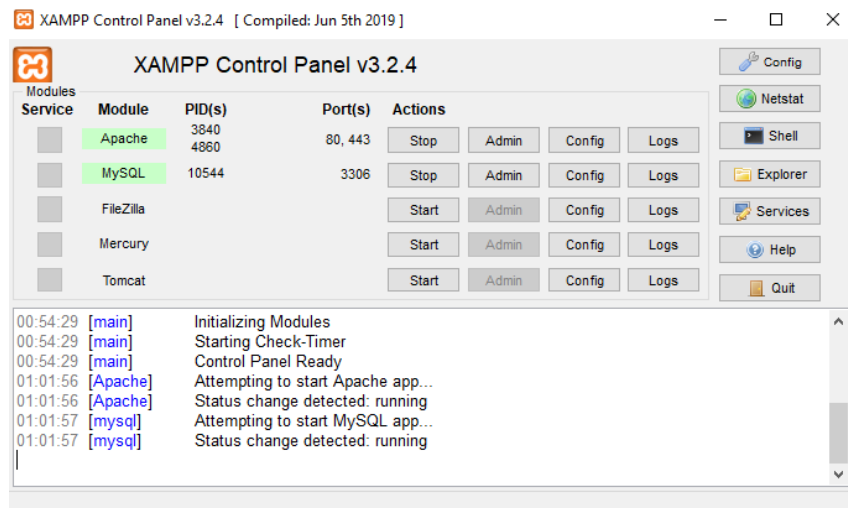
Figur 8 XAMPP installations sida

Efter nedladdningen av XAMPP ska Joomla laddas ner. Joomla laddas ner från (Joomla.org) som visas i figur 8.



Figur 9 Joomla's version

För att ladda ner Joomla behöver användaren trycka på download Joomla. Versionen beror på när experimentet utförs. I detta experiment utfördes det på en Joomla version 3.9.27 som visas i figur 9.



Figur 10 XAMPP kontrollpanel

Efter installationen Joomla överförs Joomla mappen till htdocs mappen som ligger i XAMPP mappen och sedan startas Apache och MYSQL i kontrollpanelen som i Figur 10. Efteråt går det att navigera sig till mappen med att skriva localhost/filnamnet.

Efter navigationen till Joomla webbsidan ber Joomla att användaren skriver in information som exempelvis webbsidans namn, databas osv. Joomla skapar även ett "super" konto till användaren där det ska gå att logga in i backend för att göra alla de ändringar som krävs för att utföra detta experiment.

5.3 Progression

I detta kapitel ska det redovisas hur genomförandet av implementationen stegvis gick till. I implementationen ska två hemsidor skapas för att sedan utföra en pilotstudie. Den ena webbsidan ska skapas med ramverket React och det andra ska skapas med hjälp av CMS:et

Joomla. I båda webbsidorna ska Bootstrap användas som CSS för att göra webbsidans layout identiska.

5.3.1 Joomla template

För att skapa en Joomla template behövs det först skapas en mapp. I denna mapp ska det skapas fler andra mappar och filer. Filerna går att se i Figur 11 nedan.

Namn	Senast ändrad	Typ	Storlek
bilder	2021-03-30 00:41	Filmapp	
css	2021-03-29 16:06	Filmapp	
html	2021-03-24 17:21	Filmapp	
js	2021-03-24 18:59	Filmapp	
index.php	2021-04-05 13:52	PHP-fil	14 kB
templateDetails.xml	2021-03-24 17:21	XML-fil	2 kB

Figur 11 Joomla template

I Figur 11 visas det hur template mappen ser ut. De två viktigaste mapperna är index.php och templateDetails.xml. Resten av filerna är till för att referera till i index.php. Ett exempel är Bootstrap som ligger i CSS som refereras till i index.php för att få tillgång till de fördefinierade funktionerna.

```

<header class="p-3 bg-dark text-white">
  <div class="alert position-relative alert-danger" role="alert">
    Bokrea på 15 kr
  </div>
  <div class="d-flex flex-wrap align-items-center
justify-content-center justify-content-lg-start">
    <a href="/" class="d-flex align-items-center mb-2 mb-lg-0
text-white text-decoration-none">
      <svg class="bi me-2" width="40" height="32">
        <use xlink:href="#Bootstrap" /></svg>
      </a>
      <div class="text-end">
        <button type="button" class="btn btn-outline-light
me-2">Login</button>
        <button type="button" class="btn
btn-warning">Sign-up</button>
      </div>
    </div>
  </header>

```

Figur 12 Joomla index.php navbar

Koden i index.php ska redogöra hur hela templatens ska se ut. Ett exempel på det är i Figur 12 där en navigationsmeny skapas i index.php.

```

<files>
  <filename>index.php</filename>
  <filename>templateDetails.xml</filename>
  <folder>bilder</folder>
  <folder>CSS</folder>
  <folder>js</folder>
  <folder>html</folder>
  <filename>index.html</filename>
</files>

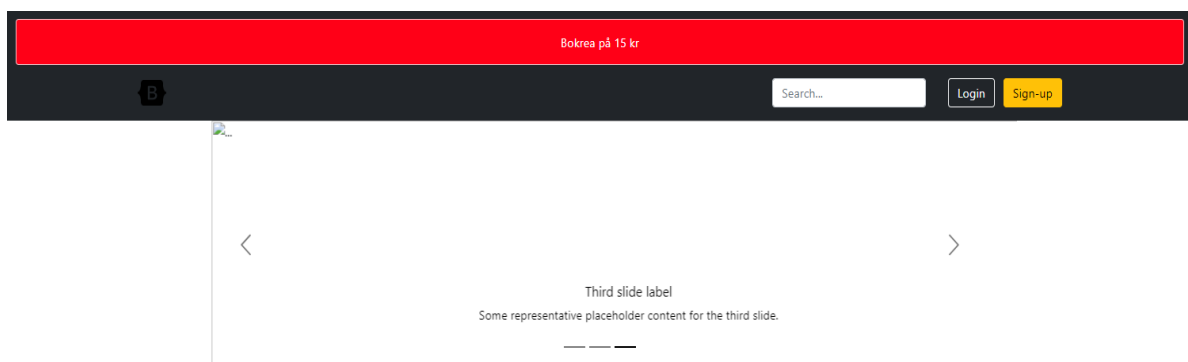
```

Figur 13 TemplateDetails.xml filer

TemplateDetails.xml är en xml fil som visar vilka filer och moduler som templatens innehåller. I Figur 13 syns det hur strukturen på filen ser ut.

Nästa steg är att ladda upp denna mapp i Joomla applikationen. Uppladdning av templatens görs i backend. För att navigera till backend behöver användaren skriva localhost/filnamn/administrator. Efteråt loggar användaren med kontot som skapades under Joomla installationen.

I backend går det senare att komma vidare till Extensions install genom att trycka på Extensions > manage. Efter det går det att installera template mappen under upload package file. För att aktivera den nedladdade filen ska användaren navigera till Extensions > templates och därefter trycka på den templates som har blivit nedladdad. Resultatet av templatens visas i figur 14.



Figur 14 Template Joomla webbsidan

5.3.2 Joomla module

Efter att skapat Joomla template måste det nu skapas en module som ska visa produkterna på webbsidan. Först installeras det en module från j2store (j2store.org). Efter att installerat modulen måste den nu också installeras i Joomla applikationen precis som templatens. Detta görs i Extensionssidan i backend. När modulen är installerad behöver den nu bli aktiverad precis som templatens. Detta görs med att navigera till extension > modules. Efter att man har hittat modulen ska användaren nu aktivera den och placera den på rätt ställe.

I TemplateDetails.xml filen står det hur många moduler som ska finnas på webbsidan och vad de heter. Här är det viktigt att placera den på rätt position på templatens. I detta fall har våran template en plats för block 1 som visas i figur 15.

```

<positions>
  <position>menu</position>
  <position>showcase</position>
  <position>block1</position>
  <position>block2</position>
  <position>block3</position>
</positions>

```

Figur 15 TemplateDetails.xml filen

```

<?php if ($this->countModules('block1')): ?>
<jdoc:include type="modules" name="block1" style="none" />
<?php endif ?>

```

Figur 16 index.php kod för att visa modulen

Efter att modulen är placerad och aktiverad ska den nu skrivas in i templatens för att den ska synas på webbsidan. Detta görs med de tre rader kod i Figur 16 . Först är den en if sats som frågar om block 1 har en module som är placerad på den. Om det är en aktiverad module på den ska den nu visas på webbsidan. För att se om modulen fungerar behövs det först installeras ett plugin som heter jstore för att kunna göra produkter. Detta görs i extension > modules och sedan install from Web.

Efter en sökning j2store kommer tillägget upp. Nu när det skapas en artikel i content > article går det att göra om den till en produkt för att sedan visa i figur 17.

PRODUCT INFORMATION

Treat as a product ☐ No ☒ Yes

Product Type **Simple**

[Change Product type](#)

Visible in storefront? ☐ No ☒ Yes

SKU

UPC/EAN/JAN/ISBN

Brand or Manufacturer

Tax Profile

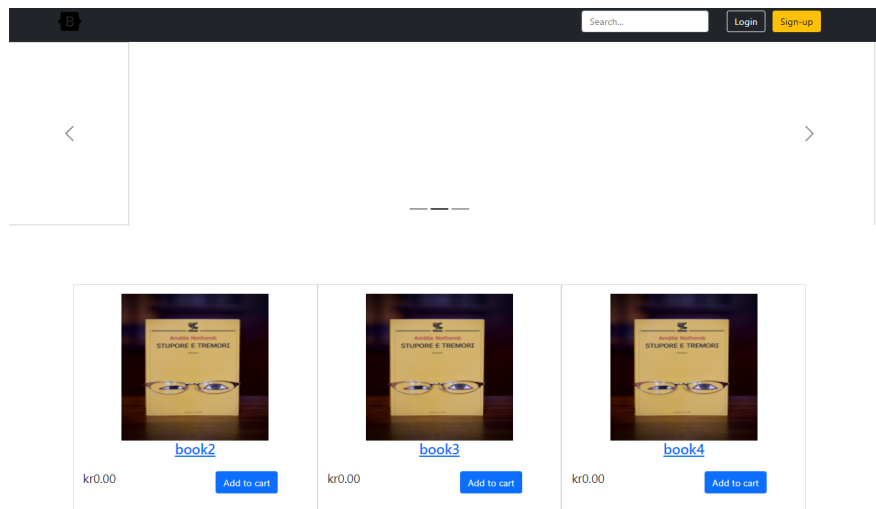
Main Tag

Cart button text

Product css class

Figur 17 Skapande av produkter

Efter att produkten blivit publicerat i templatet visas den nu med hjälp av modulen. Resultatet av produkterna av modulen visas i figur 18.



Figur 18 Module resultatet

Efter att modulen fungerar går det att välja vilken template som ska visa produktvyn när användaren trycker på den. Detta går att ändra på extension> template som visas i figur 19. ¹

¹<https://GitHub.com/a18abuah/examensarbete/commit/b1b4241efd37aef76ff792637681baff0863ee2>

Style Name *

Details Advanced **Menu Assignment**

Menu Selection:

☒ Toggle All Selections

☒ Toggle Selection

HikaShop default menus

- ☐ Categories listing
- ☐ Products listing
- ☐ User control panel
- ☐ Registration form
- ☐ Brands listing

☒ Toggle Selection

User Menu

- ☐ Your Profile
- ☐ Submit an Article
- ☐ Site Administrator
- ☐ Template Settings
- ☐ Site Settings

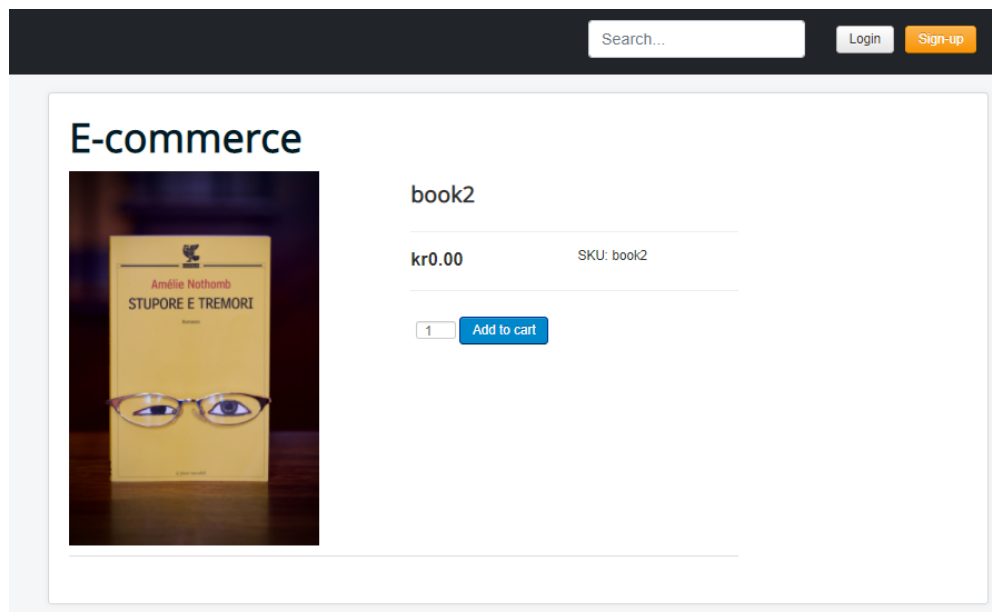
☒ Toggle Selection

Main Menu

- ☐ Home
- ☐ about
- ☐ - book
- ☒ shopping

Figur 19 Produktvy template

Det går att skapa en ny template för produktvyn men i detta arbete användes det en redan existerande template för att spara tid och bara ändra CSS för att få den lik index.php. Resultatet av produktvyn av produkten syns i figur 20.



Figur 20 Produktvyn

5.3.3 React render

Första steget med att implementera en React applikation är att skapa en html fil som det visas figur 21. I denna html fil ska det länkas två React filer. Båda filerna går att installera ifrån (React.org 2021) webbsida. Efter det ska även en div läggas till där alla React

komponenter ska renderas. Tillslut ska även ett JavaScript fil skapas där React komponenter skapas.

```
<div id="root"></div>
<script src="React.js"></script>
<script src="React-dom.js"></script>
<script src="app.js"></script>
```

Figur 21 Index.html

I app.js ska olika React element skapas som sedan ska visas i index.html div:en. I app.js skapas först en class som extendar ifrån React.component. I denna class skrivs det ut en render funktion. I denna render funktion returneras alla komponenter som ska skrivas ut på index.html. Komponenterna skrivs ut med att skriva React.component och sedan element namnet.

```
React.createElement("div", {
  className: "alert position-relative alert-danger",
  role: "alert"
})
```

Figur 22 app.js createElement

I Figur 22 presenteras det hur ett element blir skapade i render funktionen. I detta exempel skapas det en div med en klass och en role.

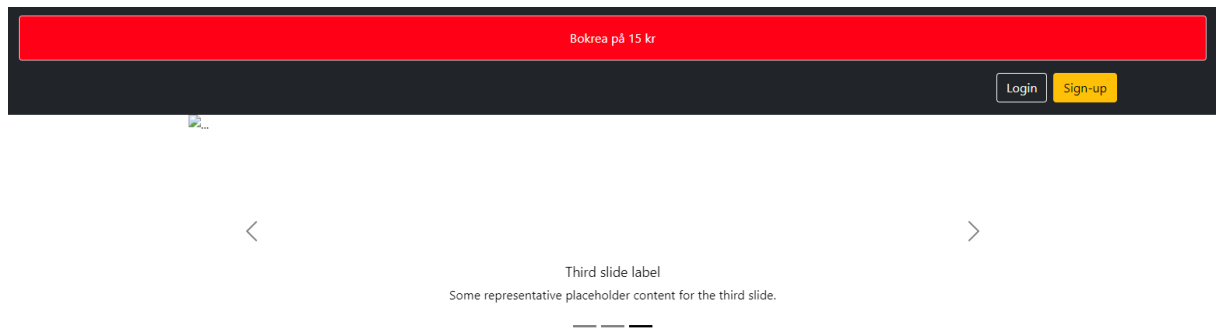
Detta görs med navbaren, footer, bild-karusellen och även sidomenyn. När alla dessa klasser har element i sig ska de senare in i en render funktion som ska visa ut alla klassers root div:en som ligger i index.html.

```
ReactDOM.render( React.createElement("div", null,
React.createElement(navbar, null),
React.createElement(carousel, null),
React.createElement(meny, null),
React.createElement(footer, null),
document.getElementById('root')));
```

Figur 23 app.js ReactDOM.render funktion

Detta görs med hjälp av React.dom.render som det visas i Figur 23. I denna render funktion skickas hela klass innehållet till index.html filen. Det är även viktigt att skapa en div över alla

element för att render funktionen ska fungera. Resultatet visas i figur 24 där reacts layout visas.²



Figur 24 React index.php resultat

5.3.4 React JSON

Efter att template för webbsidan blev skapad var det dags att skapa en mer dynamisk sida. Med att skapa en dynamisk React sida kommer det gå att lägga till men även ta bort böcker vid behov utan att skriva till extra kod. Detta görs med hjälp av JSON. Först måste en CSV fil skapas för att sedan kunna göra om den till en JSON fil. Detta görs med hjälp av ett skript. Det skriptet gör är att skriva ut id, class, bildkälla och navigering till produktvysidan i en for loop som det visas i figur 25.³

²<https://GitHub.com/a18abuah/examensarbete/commit/9bod3df98fe3f5d992aa67b644440a0ee4e15149>

³<https://GitHub.com/a18abuah/examensarbete/commit/85b7769dbddf9517c4fecdf8183933e8375b4dc3>

```

text +=
    name +
    "," +
    id +
    "," +
    idprodukt +
    "," +
    href +
    "," +
    img +
    "," +
    kr +
    "<br>";

```

Figur 25 CSV skriptet

I detta arbete skrevs det ut 100 böcker med olika information. Efter att genererat CSV filen används det en compiler med hjälp av webbsidan (CSVJSON.com 2021). Det funkar genom att först skriva in de olika rubrikerna som namn, id, idprodukt, href, img och kr och sedan klistras det in CSV filen. Efter det genererar webbsidan ut JSON fil med de värdena som står i CSV filen. Efter att JSON filen är skapad ska den namnges och skrivs in i app.js. För att få ut värdena ur JSON filen används det map() funktionen i JavaScript som visas i figur 26.

```

data1.map((dat, key) => {

```

Figur 26 Map() funktion i app.js

Funktionen som visas i figur 26 tar värdena ur data1 som är JSON filen och gör sedan en ny array med värdena ur JSON filen. För att komma åt JSON filens data går det att skriva dat som är arrayen skapad i figur 26 och sedan namnet på delen av json filen som ska skrivas ut. Detta visas i figur 27 där name delen av JSON filen skrivs ut.

```

React.createElement("div", {
    itemProp: "itemListElement",
    itemScope: "",
    itemType: "https://schema.org/ListItem",
    className: dat.name
})

```

Figur 27 map() funktion i en div

Här blir div skapad för varje "name" som finns i JSON filen. Om JSON filen innehåller 20 olika name kommer 20 div bli skapade i funktionen. Detta görs med alla produkter för att inte behöva hårdkoda varje produkt för hand. Det blir även mycket enklare att navigera mellan sidorna och mäta svarstiden som görs senare i experimentet.⁴

För att göra det enklare att mäta svarstiden skapas det även en produktvy för React. Svarssidan använder sig utav samma JSON fil för att skapa lika många produkter i produktvyn som det finns produkter på webbsidan. Skillnaden är att produkterna i produktvysidan har mer information om boken precis som produktvyn i Joomla webbsidan. I produktvyn syns bara den produkten användaren tryckt på. Detta betyder då om en användare exempelvis trycker på bok 1, då ska bara bok 1 synas på produktvyn.

```

handleClick(event) {
    const id = event.target.id;
    if(id){
        reply_click(id);
    }
}

```

Figur 28 handleClick i app.js

Detta görs med handleClick funktionen som visas i figur 28. Varje gång en användare trycker på produkten aktiveras det en handleClick. I handleClick funktionen sätts en annan funktion i gång med inparametern av id som användaren tryckt på. Denna funktion visas i figur 29.

⁴<https://GitHub.com/a18abuah/examensarbete/commit/5c36c24ef6e2563a875c8eaa18549ae4b27f073b>

```
function reply_click(id){  
  localStorage.setItem("id", id);  
}
```

Figur 29 reply_click funktion i app.js

I denna funktion sätts id som användaren har tryckt på till localStorage under värdet id. När detta värde är satt kommer svarssidan endast visa den produktvyn med det id. Detta görs genom två rader kod som visas i figur 30.

```
var id50 = localStorage.getItem("id");  
document.getElementById(id50).style.display = "block";
```

Figur 30 Produktvy sidan

De två rader koderna får först ut värdet från id till en variabel. Sedan visas bara produkten med det id som användaren har tryckt på. Denna kodsnuitt placeras under renderingen för att produkt ska kunna först skapas för att sedan visa den.⁵

5.3.5 Automatisering skript

Vid mätningar av svarstiderna av båda hemsidorna måste det gå att skapa många produkter på ett automatiskt sätt utan att behöva hårdkoda eller skapa nya artiklar för hand i Joomla backend.

Det går att uppnå detta i React genom att skapa en större JSON fil med hjälp av CSV skriptet⁶ och sedan göra en större JSON fil med hjälp av (CSVJSON.com 2021). Desto mer produkter det finns i JSON filen ju mer produkter kommer React webbsidan innehålla och detta går att välja i CSV skriptet.

I Joomla är det inte lika enkelt och därmed kräver ett skript som skapar nya produkter på ett snabbt och smidigt sätt. I detta arbete skapades det med hjälp av två Tampermonkey (Tampermonkey.net 2021) skript. Tampermonkey är ett userskript som går att lägga till i webbläsare för att sedan köra skript direkt på en specifik webbsidan.

Automatiseringsskriptet körs på Joomlas "new article" där artiklar skapas. Skriptet visas i figur 31. Först skrivs titeln och text i textfältet. Namnet på titeln kommer från en JSON-fil. I detta arbete används det samma JSON fil som i React webbsidan för att kunna göra båda webbsidorna identiska.

⁵<https://GitHub.com/a18abuah/examensarbete/commit/86cb15cdf8d2dca51450fe6fa75a4c269d57de4c>

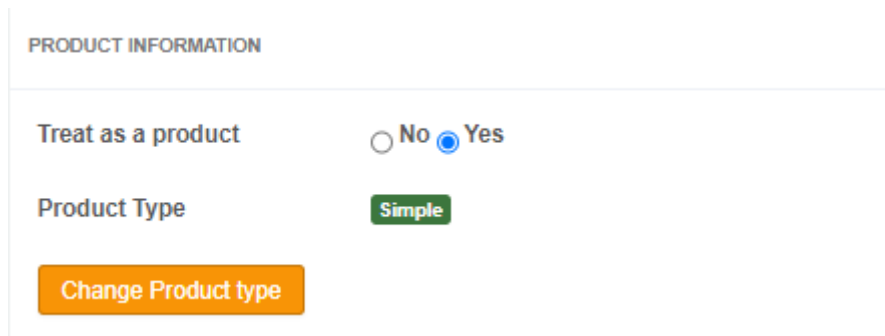
⁶<https://GitHub.com/a18abuah/examensarbete/commit/85b7769dbddf9517c4fecdf8183933e8375b4dc3>

```
var data = data1[timesIterated];
var trim = datablog.name.trim();
document.getElementById('jform_title').value = trim;
```

Figur 31 JSON fil i Joomla skript

Först skapas det en variabel med JSON-filens data med inparameter timesIterated. Variabeln timesIterated innehåller de antal gångerna skriptet körs. Efter det skapas det en annan variabel som bara visar “name” datan i JSON-filen. Därefter sätts variabeln som titeln för Joomla artikeln. För varje gång detta skript körs kommer den att först namnge titeln sedan gå ner i listan av namn i JSON-filen med hjälp av inparameter variabeln.

Efter det ska artikeln göras till en produkt. Detta görs genom att trycka på ett ja alternativ och sedan trycka på “change product type” knappen i figur 32. Detta utförs i kod exemplet på figur 33.



The screenshot shows a Joomla article edit form. Under the 'PRODUCT INFORMATION' section, there are two options: 'Treat as a product' with radio buttons for 'No' and 'Yes' (where 'Yes' is selected), and 'Product Type' with a dropdown menu showing 'Simple'. Below these options is an orange button labeled 'Change Product type'.

Figur 32 Joomla new article sida.

```
varclickelement=
document.getElementById('j2store-product-enabled-radio-group1');
clickelement.click();
Joomla.submitbutton('article.apply');
```

Figur 33 Joomla skript

Först skapas det en variabel med namnet clickelement. Denna variabel kommer sedan få värdet av “yes” alternativet av Figur 32. Efter det ska variabeln nu klickas med hjälp av click() funktionen. Därefter ska “change product type” tryckas. Detta görs med Joomla funktionen Joomla.submitbutton med id av knappen som inparameter..

Därefter ska det andra skriptet köras. Det andra skriptet är till för att ge varje produkt en bild. Anledningen till varför detta görs i ett annat skript är för att det inte fungerade att skapa en produkt och ge den en bild samtidigt.

```
document.getElementById('jform_image_thumb_image').value = image1;
document.getElementById('jform_image_main_image').value = image1;
document.getElementById('input-thumb-image').src = image1;
document.getElementById('input-main-image').src = image1;
```

Figur 34 Joomlas andra skript

I Figur 34 visas det hur bilderna på webbsidan blir tillagda genom att först skapa en variabel med en bild källa. Sedan läggs värdet av variabeln in i de i både produktvyn och webbsidan genom document.getElementById. Det krävs även att bildkälla är inskriven för att bilden ska synas på webbsidan.⁷

5.3.6 Svarstid-mättnings skript

Då båda teknikerna renderar webbsidan på olika sätt krävs det olika skript för att mäta svarstiderna på respektive sida.

5.3.6.1 React

React använder sig utav rendering och därmed blir det svårare att veta när webbsidan har renderat klart. React har en inbyggd funktion som heter componentDidMount() som visas i figur 35. Denna funktion blir tillkallad varje gång komponenterna har renderats klart. Det är med hjälp av den funktion som svarstiden ska bli uträknad.

```
componentDidMount() {
  let str1 = localStorage.getItem("theData1");
  let measurement1 = new Date();
  measurement1 = measurement1.getTime();
  str1 = "Start of textfile: \n";
  str1 = localStorage.getItem("theData1");
  str1 += measurement1 + "\n";
  localStorage.setItem("theData1", str1);
  window.location.assign('http://127.0.0.1:5500/index.html');
}
```

Figur 35 componentDidMount i produktvyn

I figur 35 sparas det ett datum i en variabel med date() funktionen. Efter det blir datumet omvandlat till millisekunder med getTime() funktionen. Detta värde blir sparad i localStorage för varje gång användaren trycker på produkten. Därefter navigerar den tillbaka till webbsidan för att skapa en loop.

⁷<https://GitHub.com/a18abuah/examensarbete/commit/803ba00b11088daa67a110d99d440c7c33b1df95>

Efter det behövs det även ett annat skript som navigerar till de olika produkterna. Uppgiften med andra skriptet är att både navigera till produktsidan och spara värde i millisekunder så det värdet kan subtraheras med värdet från componentDidMount funktionen. Det skriptet körs i Tampermonkey (Tampermonkey.net 2021) tillägget som i Joomla automatiseringens skriptet.

```
let measurement = new Date();
measurement = measurement.getTime();
if (count > iterations) {
    localStorage.setItem("Counter", 0);
    localStorage.setItem("counter1", 0);
} else {
    if (isNaN(count)) count= 0;
    if (count == 0) {
        str3 = "Start of textfile: \n";
        localStorage.setItem("Oldval", measurement);
    } else {
        str3 = localStorage.getItem("olddata");
        str3 += measurement + "\n";
    }
}
```

Figur 36 React mätningsskript

I Figur 36 skapas det att en variabel med värde likt det i componentDidMount funktion. Värdet skrivs endast ut i localStorage om iterationerna är större än count. Count är en variabel som går upp i värde för varje gång funktionen utförs och iteration är maxvärde för de antal mätningarna som ska utföras.

```
var datablog = data1[counter];
var trim1 = datablog.id.toString();
localStorage.setItem('id', trim1);
window.location.assign('http://127.0.0.1:5500/index2.html');
```

Figur 37 React mätningsskript navigation

I Figur 37 visas det hur React mätning skriptet navigerar mellan produkterna. Först skapas det en variabel som tar JSON värdet. JSON värdet har även counter variabeln som inparameter. Count variabeln har det värdet som stiger för varje gång funktionen körs. Efteråt skapas det en ny variabel som först tar id värdet och sedan gör om det till string. Sedan blir det id värdet tillagt i localStorage för varje gång funktionen körs. Tillslut navigerar

skriptet sig till produktvy sidan med hjälp av `location.assign()` funktionen. Det som händer i produkt vyn är att den produkten med id som blir tillsatt i skriptet visas och sedan går den ner i JSON filen tills counter variabeln värde blir lika mycket som iterations variabelns värde och då stoppas funktionen.⁸

Efter att både värdena blir utskrivna i `localStorage` ska värdet ifrån skriptet subtraheras med värdet från `componenDidmount()` funktionen. Detta kan göras med med ett google sheet funktionen `=minus()`. Det värdet som kommer ut blir då delta värdet som ska jämföras med Joomlas delta värde.

5.3.6.2 Joomla

Skillnaden mellan svarstidsmätningen i Joomla och React är att Joomla svarstid kan räknas ut med endast ett skript. Till skillnad från React behöver inte Joomla någon funktion som blir tillkallad efter att sidan blir renderat. Därmed går det att räkna ut delta värdet direkt i skriptet. Joomla skriptet visas i figur 38.

```
let measurement = new Date();
measurement = measurement.getTime();
if (cnt > iterations) {
    localStorage.setItem("Counter", 0);
    localStorage.setItem("Counter1", 0);
} else {
    if (isNaN(cnt)) cnt = 0;
    if (cnt == 0) {
        str = "Start of textfile: \n";
        localStorage.setItem("Oldval", measurement);
    } else {
        var old = new Date();
        old = old.setTime(localStorage.getItem("Oldval"));
        var delta = measurement - old;
        str = localStorage.getItem("theData");
        str += delta + "\n";
    }
}
```

Figur 38 Joomla mätningsskript

Den enda skillnaden mellan de olika mätningsskripten är att i Joomlas blir ett nytt värde tilldelat i funktionen efter att skriptet har navigerats till produktvyn. Det värdet blir subtraherat med värdet innan sidan navigerade och därmed skrivs delta värdet ut. Efteråt

⁸<https://GitHub.com/a18abuah/examensarbete/commit/718098878b23b54547848a24aa945f118de19108>

blir delta värdet sparas i localStorage. Efter att värdet blir sparad måste det ske en navigation mellan sidorna. Detta sker genom raderna kod som visas i figur 39.

```
var datablog = data1[counter];  
var trim = datablog.name.trim();  
// ReLoad page!  
  
window.location.assign("http://localhost/Joomla3.9/index.php/component/j2  
store/products/" + trim);
```

Figur 39 Joomla mätningsskript navigation

Navigationen på de mätningsskripten är också likadana. Den enda skillnaden är att React har en produktvy sida som visar och döljer olika produkter beroende på vilken produkt användaren trycker på. Joomla däremot skapar en egen sida för varje produkt. Detta är därför name blir trimmat i JSON filen istället för id som sedan blir tillagd i slutet av webbsida källan.⁹

Tillslut ska en loop skapas med hjälp av ett annat skript som körs på produktvyn. Detta utförs med raden kod som visas i figur 40.

```
window.location.href = "http://localhost/Joomla3.9/";
```

Figur 40 Joomla navigation tillbaka till skript

Det enda som sker i skriptet är att den navigerar tillbaka till webbsidan där skriptet körs. Med hjälp av detta skript skapas det en loop som körs tills det andra skriptet är färdigt.¹⁰

⁹<https://GitHub.com/a18abuah/examensarbete/commit/4f7b7739331312d6a4efc19dd3f7866e62b86b78>

¹⁰<https://GitHub.com/a18abuah/examensarbete/commit/712b19a316a7e07b79bd708437261347ad58f6ad>

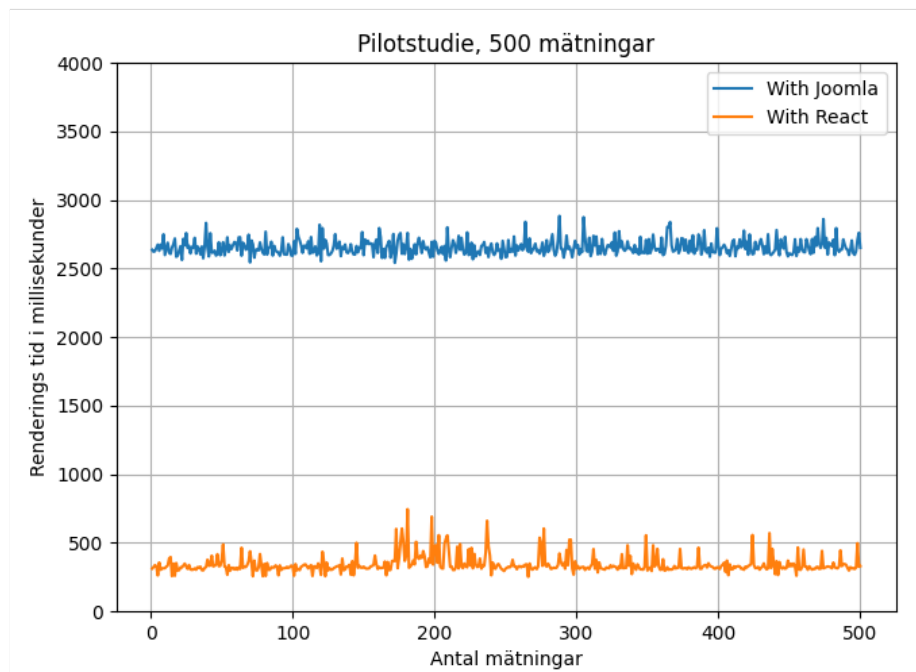
6. Pilotstudie

I detta arbete genomfördes det en pilotstudie. Pilotstudiens uppgift är att se om mätningarna som utförs i arbetet mäter korrekt och är genomförbart. Denna pilotstudie går ut på att ett skript ska navigera mellan produktvy och webbsidan. Navigationen sker genom att skriptet först itererar till produktvy och sedan tillbaka till webbsidan. Under tiden navigationen sker kommer data sparas i lokalstorage som indikerar hur lång svarstiden var. Varje navigation räknas som en mätpunkt och båda hemsidorna navigationen sker identiskt. Detta betyder att när book 1 vyn i React navigeras till först ska även detta hända i Joomla webbsidan så varje produkt jämförs med varandra i mätningen. Det används två olika skript för React och Joomla. Den enda skillnaden mellan skripten är att React har en speciell teknologi för rendering och därmed behöver skriptet räkna datapunkten efter renderingen medan Joomla går att räkna direkt i webbläsaren. I denna pilotstudie ska 500 mätpunkter mätas på båda hemsidorna. Dessa 500 mätpunkter visas i ett linjediagram i Figur 41 och stapeldiagram i Figur 42 som millisekunder.

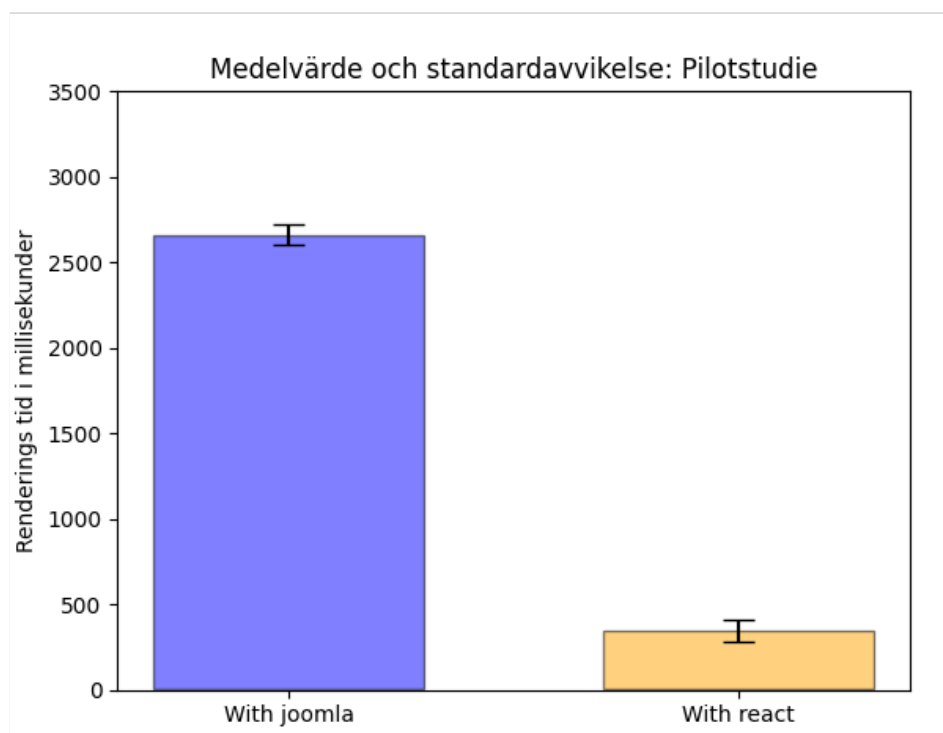
Mätningen sker lokalt för båda hemsidorna. Pilotstudien ska även utföras på samma dator och samma nätverksuppkoppling för att få samma förutsättningar. Mätpunkterna sparas på localstorage i Chrome för båda testerna. Hårdvaran och mjukvaran som experimentet utfördes på går att se på tabell 1 nedan.

Tabell 1 Specifikationer

Processor	Intel(R) Core(™) i5-4690K CPU @ 3.50GHz
RAM	8GB
Grafikkort	AMD Radeon (™) R9 380 Series
Operativsystem	Windows 10 Pro N 64 Pro N 64 bitar
Webbläsare	Version 89.0.4389.128



Figur 41 linjediagram för pilotstudie



Figur 42 stapeldiagram för pilotstudie

6.1 Diskussion av pilotstudie

I pilotstudien är skillnaden mellan React och Joomla svarstid väldigt tydligt. I linjediagrammet på Figur 41 syns det inga spikar i datan som mäts på både React och Joomla. Detta indikerar att navigera mellan sidorna tog ungefär lika mycket tid utan några störningar. Skillnaden är också väldigt tydlig på stapeldiagrammet i Figur 42 där medelvärdet på React ligger på 480 ms medan Joomla ligger på 2700 ms. Standardavvikelsen på stapeldiagram är även liten på båda mätningar och den överlappar inte vilket indikerar att det finns en skillnad mellan de två testerna. Då pilotstudien inte har indikerat några spikar eller diverse problem i mätningen anses den vara lyckad. En grej att uppmärksamma är att React har en större standardavvikelse än Joomla. Detta kan bli intressant att mäta på i testfallet för att se om Joomla kan hålla den lilla standardavvikelsen och om Reacts standardavvikelse kommer att öka med hur många mätpunkter som läggs till. Förbättringen inför testfallet är att öka mätningen för att kunna dra en slutsats om vilken teknik som har den bästa svarstiden för en e-bokhandel webbsida. En annan förbättring är att öka antalet data och sedan jämföra det mellan Joomla och React för att se vilken teknik som fungerar bättre med större projekt.

7. Utvärdering

7.1 Presentation av testfall

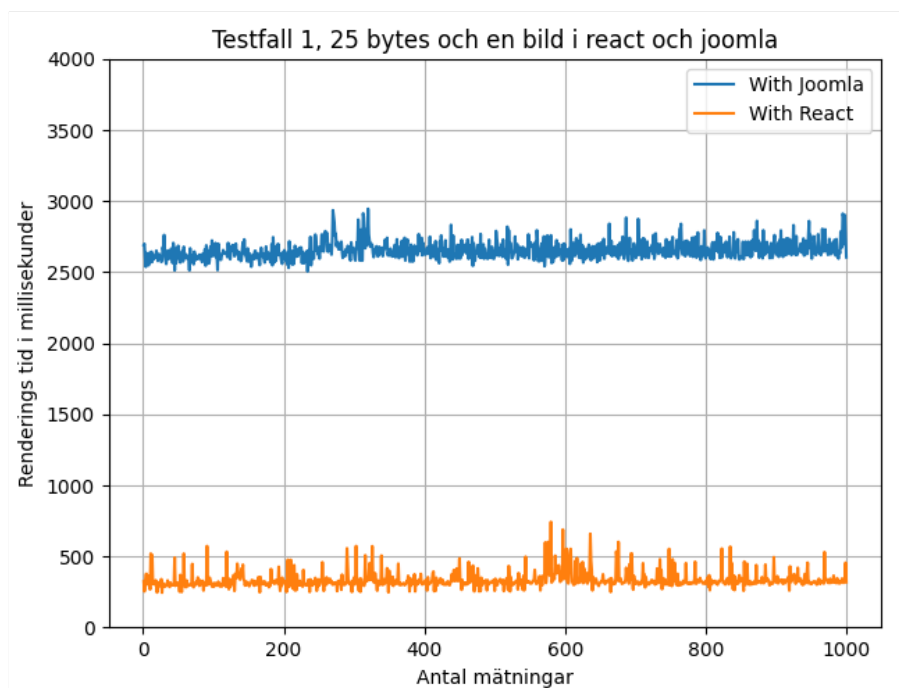
Efter en lyckad pilotstudie ska det utföras fyra testfall för att kunna dra en slutsats och diskutera kring arbetets hypotes. Det första testfallet innehåller en mindre mängd data som i detta arbete är upplagt på samma sätt som i pilotstudien. Den enda skillnaden är att antalet mätningar har ökat. I de mätningar som utförs på en mindre datamängd ska det utföras 1000 mätningar istället för 500 för att få ut mer data att kunna diskutera och dra slutsatser kring. Navigationen sker på 100 produkter med 24 bytes data och en bild i varje produkt. Dessa mätningar sker genom navigation mellan produkter. Det andra testfallet kommer även det efterlikna pilotstudien och det första testfallet. Den enda skillnaden är att mer data kommer läggas in i varje produkt innan mätningen. Ökningen sker genom att lägga in mer text i JSON-filen som skapar produkterna. Datan hämtas från [lipsum.com\(2021\)](https://lipsum.com/) som skapar genererad exempeltext. I testfallet ska det finnas 500 bytes data och en bild i varje produkt. Detta testfall utförs för att kunna jämföra vilken av dessa tekniker som håller bäst prestanda vid hantering av större produkter innehållandes mer information. I det tredje testfallet kommer antalet produkter på webbsidan ökas från 100 till 500 produkter. Datan kommer fortfarande vara 500 bytes och en bild på varje produkt som i det andra testfallet. Detta testfall ska utföras för att kunna jämföra vilken teknik som är bäst anpassad för en större webbsida. Fjärde testfallet kommer utföras på Firefox istället för Chrome. Bytet av webbläsare kommer visa om någon förändring sker beroende på vilken webbläsare användaren använder. Testet kommer utföras på 500 produkter med 500 bytes data och en bild i varje produkt som i testfall 3. Alla testfall och beskrivningar går att se på tabell 2.

Tabell 2 Testfall

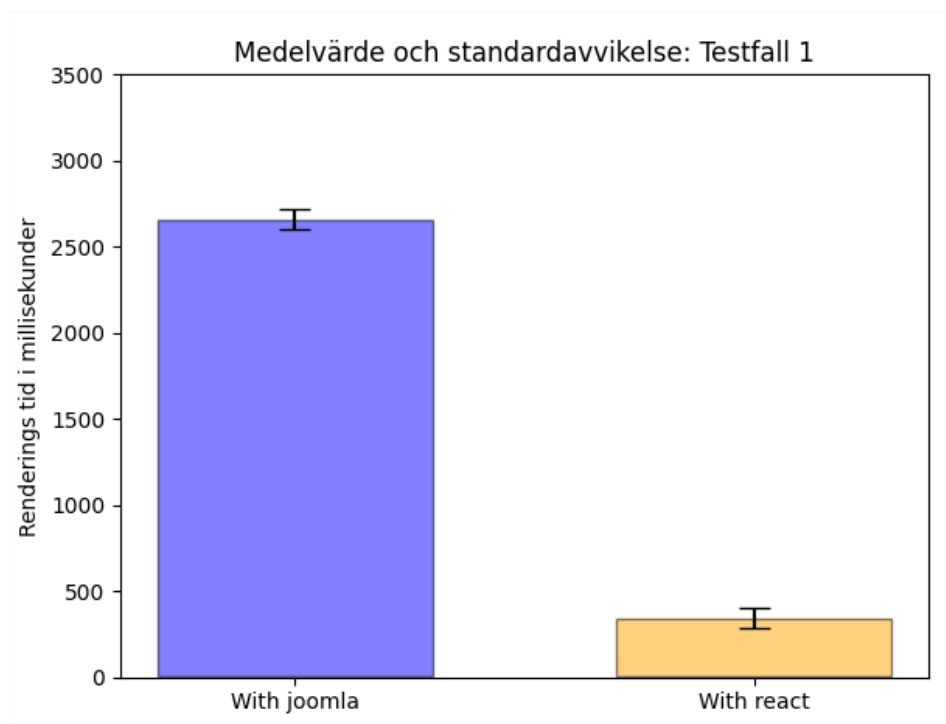
Test	Beskrivning
Testfall 1	24 bytes och en bild i varje produkt
Testfall 2	500 bytes och en bild i varje produkt
Testfall 3	Ökning från 100 till 500 produkter
Testfall 4	Byte från Chrome till Firefox

7.2 Analys av testfall

7.2.1 Testfall 1



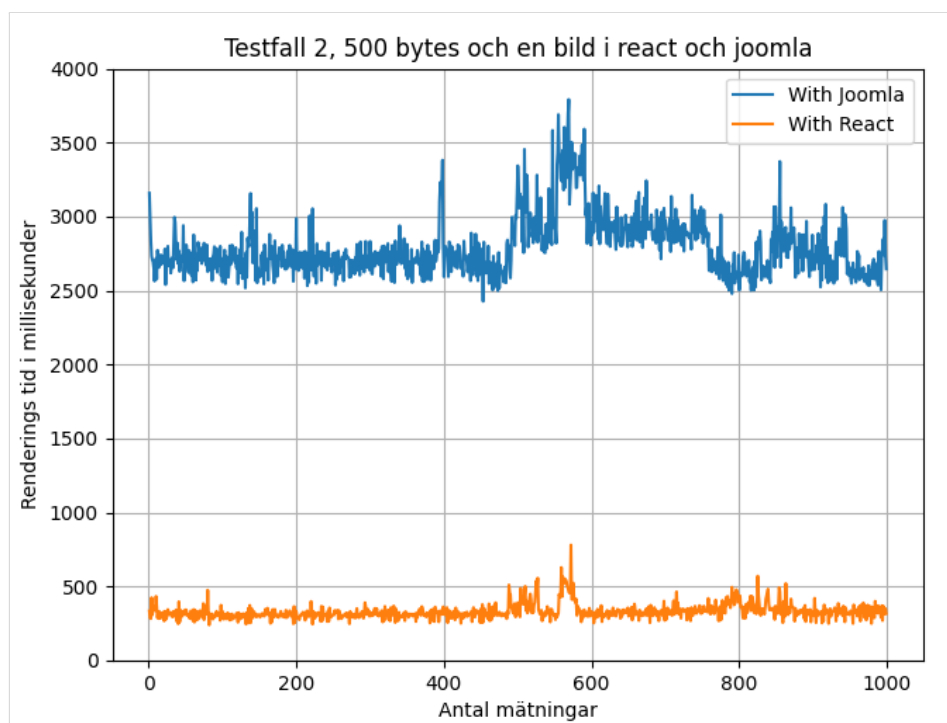
Figur 43 Linjediagram av testfall 1 med 25 bytes och en bild



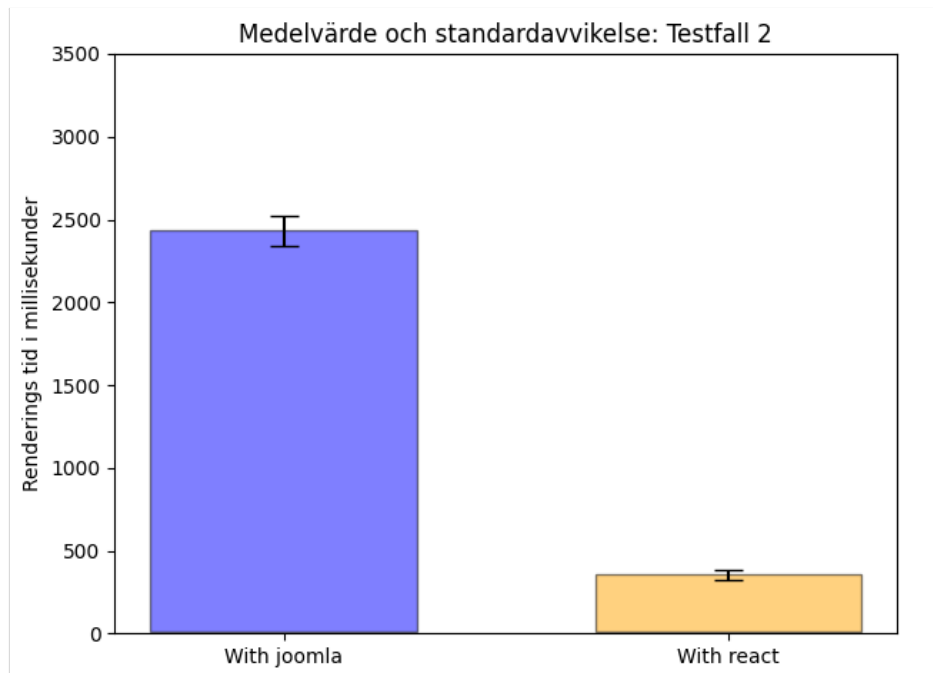
Figur 44 Stapeldiagram av testfall 1 med 25 bytes och en bild

I testfall 1 mäts svarstiden på navigationen mellan produkterna på React och Joomla webbsidan. Det utförs 1000 mätningar på båda webbsidorna som sedan jämförs med hjälp av linjediagrammet i Figur 43 och stapeldiagrammet i Figur 44. I Figur 43 syns det fortfarande en stor skillnad på svarstiderna mellan teknikerna som det gjorde i pilotstudien. Det förekommer däremot inga spikar som skulle indikera på störningar i mätningen i Figur 43. I Figur 44 syns stapeldiagrammet som visar medelvärdet för båda teknikerna. Där syns det återigen samma skillnad som det gjorde i pilotstudien. Reacts medelvärde ligger precis under 500 millisekunder medan Joomla ligger på 2600 millisekunder. Standardavvikelsen överlappar inte på stapeldiagrammet vilket indikerar att det finns en skillnad mellan mätningarna och därmed behövs inget Anova-test köras.

7.2.2 Testfall 2



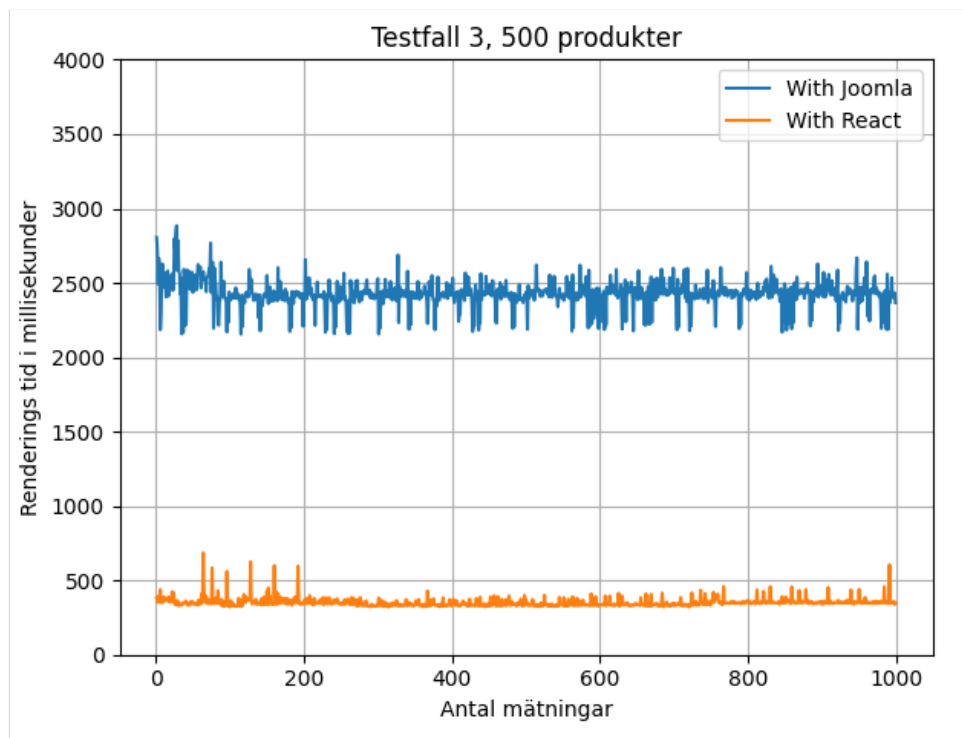
Figur 45 Linjediagram av testfall 2 med 500 bytes och en bild



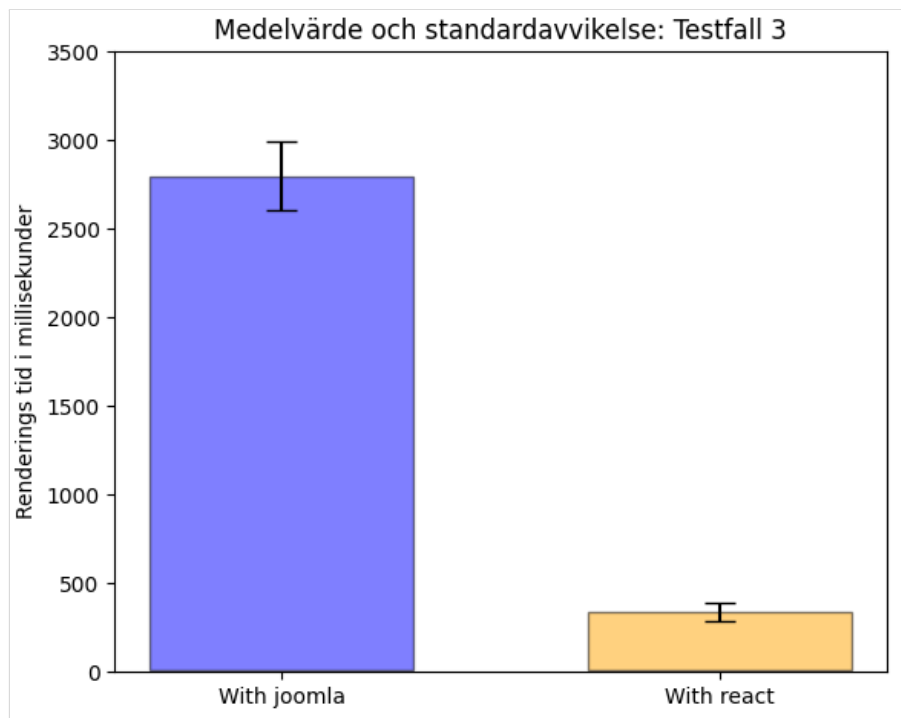
Figur 46 Stapeldiagram av testfall 2 med 500 bytes och en bild

I testfall 2 mäts svarstiden på navigationen på Joomla och React webbsidan med mer data i varje produkt. Varje produkt består av 500 bytes data och en bild jämfört med testfall 1 som bara har 24 bytes data. Resultatet av mätningarna visas i Figur 45 i linjediagrammet och Figur 46 i stapeldiagrammet. I linjediagrammet syns det att Joomla's svarstid går upp och ner jämfört med React som håller samma svarstid. I stapeldiagrammet syns medelvärdet för både React och Joomla. Reacts medelvärde ligger på strax under 500 medan Joomla ligger på 2400. Skillnaden är att standardavvikelsen är mycket större hos Joomla än React. Även i denna mätning överlappar inte standardavvikelsen och därmed behövs det inte ett Anova test för att avgöra om det är någon skillnad.

7.2.3 Testfall 3



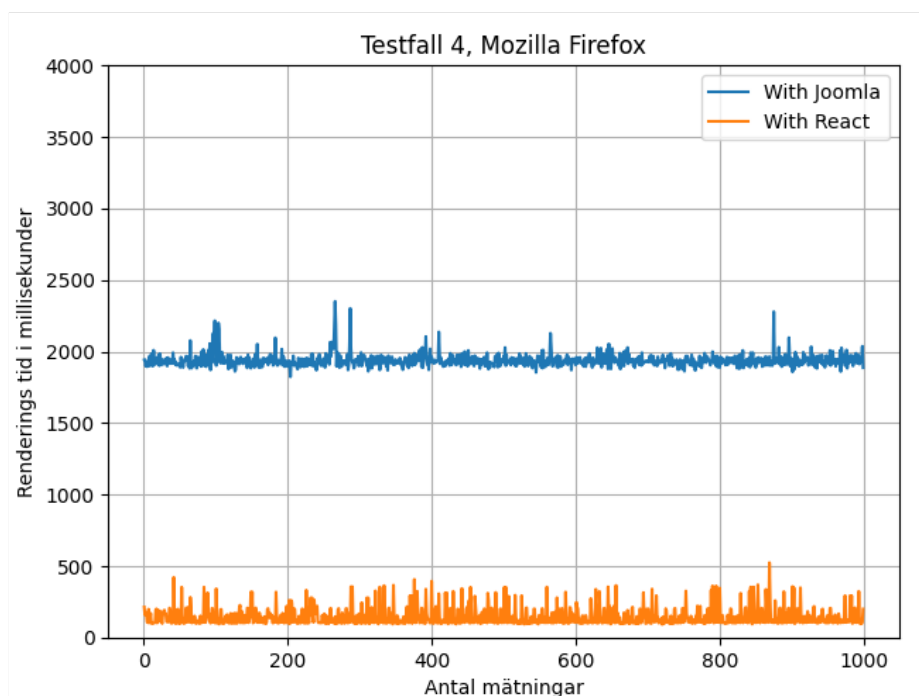
Figur 47 Linjediagram av testfall 3 med 500 produkter



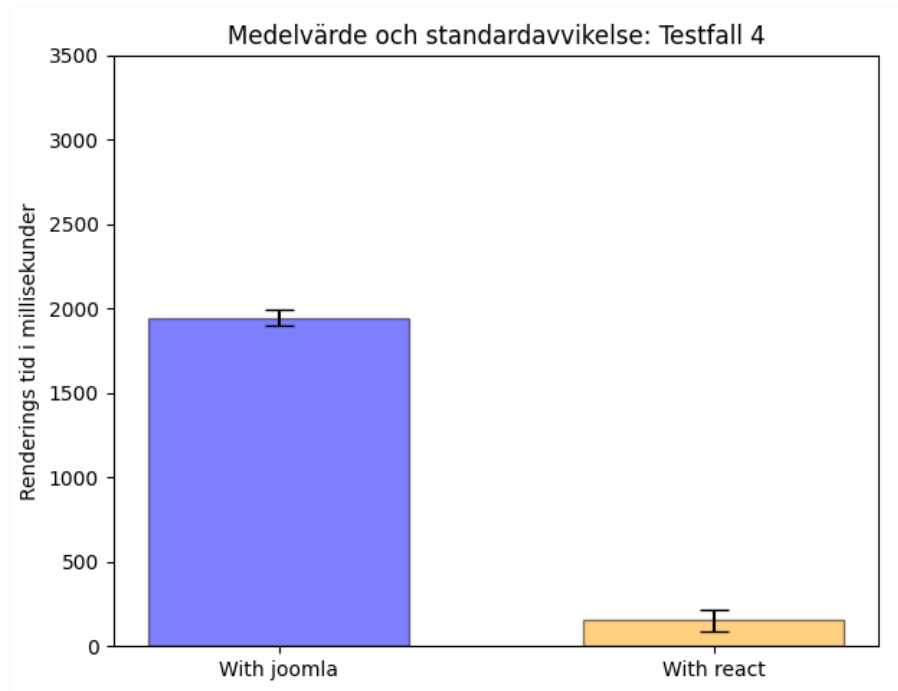
Figur 48 Stapeldiagram av testfall 3 med 500 produkter

I testfall 3 lades det till mer produkter i både Joomla och React innan mätningarna. I de föregående mätningarna var det endast 100 produkter som itererades igenom 1000 gånger. I detta testfall lades det till 500 produkter med samma 500 bytes data och en bild i varje produkt itereras igenom 1000 gånger. I Figur 47 och 48 syns det stapeldiagram och linjediagram. I linjediagrammet syns det återigen att svarstiden hos Joomla går upp och ner ju mer data som används som i förra testfallet medan React fortfarande håller sig på mestadels samma nivå i mätningen. I stapeldiagrammet syns det att Joomla har ett medelvärde på 2700 millisekunder och att React har medelvärdet på 450. Standardavvikelsen är återigen mycket större i Joomla än React. Även i denna mätning syns skillnaden med att standardavvikelsen inte överlappar och därmed behövs inte ett Anova-test utföras.

7.2.4 Testfall 4



Figur 49 Linjediagram av testfall 4 med annan webbläsare



Figur 50 Stapeldiagram av testfall 4 med annan webbläsare

I testfall 4 ska mätningen utföras med en annan webbläsare. De föregående mätningarna sker i webbläsaren Chrome. I detta testfall byts Chrome ut mot Firefox. Resultatet visas i Figur 49 och 50 med hjälp av ett linjediagram och ett stapeldiagram. I linjediagrammet syns det att Joomla mestadels håller sig till samma nivå medan Reacts svarstider går upp och ner. Detta syns senare i stapeldiagrammet där React har en större standardavvikelse än Joomla. Medelvärdet på svarstiderna från Firefox jämfört med Chrome är betydligt mindre. React som under de föregående testfallen varit runt 450 millisekunder är nu strax under 250 millisekunder. Joomla's medelvärde har även sänkts från den föregående mätningar som var runt 2500-2700 till 1900 millisekunder.

8. Avslutande diskussion

8.1 Sammanfattning

E-commerce har blivit mer och mer populärt med tiden och därmed har standarden för hur en webbsida får se ut ökat. En av de viktigaste faktorerna i en e-commerce är svarstiden. Svarstid är en viktig faktor i e-commerce enligt Mickens (2010).

I detta arbete skapades det två e-commerce applikationer med temat bokhandel som tillämpning. En webbsida skapades med React som är ett JavaScript-bibliotek och det andra med Joomla som är ett CMS. Dessa två tekniker skulle då jämföras utifrån svarstid. Frågeställning var vilken av dessa två implementationer som ger lägst svarstid. För att jämförelsen skulle bli rättvis var båda hemsidorna så lika varandra som möjligt. Detta gjordes med användandet av Bootstrap som är ett CSS-ramverk. Arbetet hade hypotesen att React skulle prestera bättre då CMS i vanliga fall är segare på grund av alla filer som laddas när webbsidan körs.

Tillämpningen som valdes i detta arbete var en e-commerce i form av en bokhandel lik den som skapades i forskningsartikeln skriven av Liang-fu et al.(2010). Informationen som lagras i bokhandel-webbsidan ska vara mestadels information om boken samt en bild på respektive bok. Denna tillämpnings tema passar en bokhandel men även andra e-commerce tillämpningar som exempelvis kläder.

De testfall som utfördes i detta arbete hade alla något att utesluta. Första testfallet var för att se vilken av teknikerna som var bäst anpassad för mindre hemsidor. Andra testfallet var utfört för att se vilken av teknikerna som är bäst anpassad för hantering av större produkter. Tredje testfallet var för att se vilken teknik som hanterade mycket produkter bättre.

Tillslut så var fjärde och sista testfallet till för att utesluta om webbläsaren är anledningen till resultaten. I alla testfallen hade React en mycket lägre svarstid. React hade runt 200-250 millisekunder i medelvärde under alla testfallen medan Joomla hade mellan 2000-2700 millisekunder i medelvärde. Med hjälp av resultatet av testfallen fastställdes hypotesen att React har en lägre svarstid än Joomla.

Då detta arbete inte har blivit utfört av någon forskare tidigare blir det svårt att se skillnader eller liknelser i resultatet jämfört med andra forskningsartiklar. Därmed stämde resultatet från forskningsartikeln skriven av Tomisa et al.(2019) där CMS-svarstider beskrevs som långsamma på grund av filer som måste laddas innan webbsidan visas. Detta kan vara anledningen till varför React laddade snabbare ihop med renderingstekniken som beskrivs i forskningsartikeln skriven av Xing et al.(2014).

Slutsatsen som kan dras från detta arbete är att React presterar bättre än Joomla i den implementationen som skapades i arbetet. Även fast Joomla är en mycket enklare teknik att skapa en webbsida på så är React bättre på att hämta sidan snabbare. I mätningen syns det att React var den bättre tekniken på att hantera mindre och större produkter i testfall 1 och 2. Testfall 3 visar även att React var den bättre tekniken för en större data med mer produkter. I testfall 4 visas det att React är den bättre tekniken både på Chrome och Firefox.

8.2 Diskussion

Innan implementationen samlades det information om de olika teknikernas fördelar och nackdelar. Joomla ansågs då vara enligt många forskningsartiklar ett enkelt alternativ att skapa hemsidor på. Det står även i forskningsartikeln skriven av Cao et al.(2010) att med hjälp av Joomla går det skapa interaktiva hemsidor på ett snabbt och smidigt sätt. Detta var inte fallet i detta arbete när det kom till skapandet av template. Detta kan bero på att de flesta användarna inte skapar egna templates utan istället använder redan fördefinierade templates.

Det förekom även mycket problem med Joomla när det kom till Bootstrap. Joomla har redan en fördefinierad fil med Bootstrap som inte gick att ta bort. Lösning fick bli att skriva över den filen för att kunna ändra i layouten av template. När moduler användes i Joomla förekom det även många element med samma klass eller id namn som inte gick att ändra på. Det blev då krångligt att ändra på ett element och inte alla andra.

Under implementationen av React valdes det att skapa en React miljö utan några ramverk eller bibliotek med hänsyn mot en rättvis jämförelse. Detta visade sig vara en svår implementation då de flesta guider, böcker och vetenskapliga artiklar skapade React med ramverk. Detta gjorde problemlösningen i React mycket svårare. Miljön som presenterades på de olika forumen var inte lik den som skapades i detta arbete

De två teknikerna som blir jämförda i detta arbete har inte blivit jämförda innan i en annan studie. Detta gör det svårt att utgå från en annan vetenskaplig artikel. Anledningen till varför dessa två blir jämförda är för att Joomla är det bästa CMS:et och React var det bästa JavaScript biblioteket till skapandet av e-commerce. Som det har nämnts tidigare i kapitel 3 har Joomla bara jämförts med andra CMS som i artikeln skriven av Patel et al.(2011) och React med andra JavaScript bibliotek som i artikeln skriven av Xing et al.(2014).

Det finns många faktorer som påverkar trovärdigheten av resultatet samt slutsatsen som drogs i detta arbete. Den första faktorn är om det går att replikera arbetet för att personer eller företag i framtida arbete kan replikera för att se om resultatet stämmer överens. Data som presenteras i testfallen får inte heller bli manipulerade på något sätt utan någon förklaring. Ett exempel kan vara att någon störning sker som inte har med tekniken att göra och därmed blir den spiken borttagen på linjediagrammet. I detta arbete presenteras hela arbetet på GitHub för att kunna göra det replikerbart och data från testfallen är inte manipulerade.

En annan faktor i trovärdigheten kan bero på att det finns olika skript för mätningen av de båda applikationerna. Som det redan nämnts tidigare i rapporten beror detta på att teknikerna använder olika renderingsfunktioner och därmed krävs det olika skript.

Både React och Joomla har sina fördelar och nackdelar när det kommer till skapandet av en e-commerce. Joomla beskrivs redan i förstudien innan implementationen att det var den enklare tekniken att skapa webbsidor på. React däremot visade med hjälp av resultatet av testfallen vara den bättre tekniken för att hämta data från en webbsida snabbare.

Resultatet av testfallen är dock helt beroende på de två webbsidorna som har skapats i detta arbete. Det har inte utförts en helhetsbedömning på React eller Joomla. Det resultatet som

presenterades i denna studie kan helt och hållet bero på den tillämpningen som användes i denna studie. I en annan tillämpning kan exempelvis Joomla visa sig vara det bättre alternativet utifrån svarstid.

Det är också viktigt att påpeka att hårdvaran och mjukvaran samt teknikerna som användes i denna studie blir ständigt utvecklade. Med det blir denna studies resultat mindre relevant med åren som kommer. Om några år kanske Joomla optimerar filerna eller använder en annan renderingsteknik som efterliknar Reacts.

8.3 Etik och samhälle

Den viktigaste fokuset ur det etiska perspektivet är att arbetet ska kunna replikeras. Arbetets progression går att se på GitHub(2021). Genom att se tillämpningen av arbetet går det enkelt ladda hem hela implementationen och utföra mätningarna. Det går även att ladda hem en äldre version och sedan bygga vidare på detta. Med replikeringen går det även att bygga på en äldre version för att testa andra versioner som kan ge ett annat resultat. Replikeringsbarheten ökar förtroendet i resultatet och i implementationen som visas i arbetet.

En annan etisk aspekt var att inte använda någon information eller bilder som är rättighetsskyddade. Det var därför detta arbete endast hade genererad information och bilder som inte är rättighetsskyddade.

Testfallen i detta arbete var inte helt övertäckande då det inte kördes med alla webbläsare och operativsystem. De webbläsarna som testades var Chrome och Firefox och alla test kördes i Windows. Det kan förekomma att resultat ser annorlunda ut på annan hårdvara och mjukvara. Mjukvaran och hårdvaran som användes i detta arbete beskrivs i tabell 1 i pilotstudien.

Detta arbete är väldigt aktuellt då människor ständigt köper och säljer varor dagligen på e-commerce webbsidor. Med mer konkurrens blir det viktigt för företag att hitta den bästa tekniken att skapa hemsidor på. Samhällsnyttan som presenteras ur detta arbete är att ge lärdom om vilken teknik som skapar den bästa e-commerce webbsidan utifrån svarstid.

En samhällsrisk som förekommer i denna studie är energiförbrukningen vid utvecklingen av diverse tekniker. Utvecklingstiden kommer att höjas ju svårare teknik som används vid skapandet av webbapplikationen och därmed blir energiförbrukningen högre. I detta arbete var React den teknik som generade kortast svarstid på ecommerce webbsidan, men JavaScript biblioteket har även en högre utvecklingstid än CMS:et Joomla. Med den högre utvecklingstiden blir React mer kostsamt vid utvecklingen av webbsidan. En annan risk är användandet av JavaScript-bibliotek som React. Nya JavaScript-bibliotek och ramverk blir ständigt utvecklade och de redan existerande blir uppdaterade med nya funktionaliteter för att nå en högre prestanda (Pano et al.2011). Om det valda JavaScript-biblioteket eller ramverket ligger efter i utvecklingen kan det förekomma en risk att det blir borttaget eller ersätt med en annan teknik. Detta skulle leda till att webbapplikationen skulle behöva byggas om med en annan teknik som kommer ta mycket tid och resurser.

I detta arbete jämförs två olika tekniker utifrån svarstider. Ur ett hållbarhetsperspektiv är det viktigt med snabbare svarstider då detta minskar energiförbrukningen för kunderna. Om

svarstiden är kort kommer detta resultera till att användare inte behöver använda energin av att vara inne på den webbsidan lika länge. Detta kommer i sin tur vara bra för miljön.

Även fast snabba svarstider är viktigt för hållbarhetsperspektivet kan det finnas en annan vinkel där det kan leda till mindre energibesparing. Om tekniken som används för att optimera svarstiden kräver mer utvecklare för att hålla uppe en sida kan detta bidra till mer energiförbruk. Då utvecklare möjligtvis behöver åka till jobbet kan redan detta bidra till ett större energiförbruk än om den enklare tekniken används där det krävs färre utvecklare.

8.4 Framtida arbete

Med hjälp av informationen av detta arbete går det skapa både en Joomla och React e-commerce webbsida. Det går se hur kodningen ser ut i React utan ramverken eller hur Joomla kan bli installerad lokalt utan någon webbserver. I framtida arbete går det skapa React med ramverk och sedan jämföra det med React utan ramverk för att se om det blir någon skillnad. Det går även skapa båda hemsidorna med en webbserver istället för lokalt som det utfördes i detta arbete.

De testfall som har genomförts i detta arbete har alla uteslutit var sin punkt som exempelvis vilken teknik som är bäst med lite data eller mer data. Det går dock fortfarande att fortsätta med de testfallen för att kunna få fler frågor uteslutna. En av de kan vara att generera data i böckerna som är verklighetsbaserade. I testfallen som utfördes i detta arbete användes det genererad data i böckerna för att slippa etiska problem och upphovsskyddat information.

I varje testfall fanns det även en mängd data i varje produkt som skulle se identisk ut i varje bok. Problemet med detta är att det inte är verklighetsbaserat. Bokhandel-webbsidor har olika mycket information. I framtida arbeten går det att skapa en bokhandel-webbsida med olika mycket information för att se om det resulterar i ett annat resultat från testfallen. Ett annat testfall kan vara att öka variationen på böckerna till ett större antal. I detta arbete var testfallen mellan 100 och 500 böcker. I framtida arbeten skulle det gå att testa att nå upp till 100 000 böcker som en vanlig bokhandel-webbsida. I de testfall som skapades i detta arbete var en av de att byta webbläsare från Chrome till Firefox. Detta var för att se om Chrome var anledningen till resultatet eller om resultatet skulle vara likadant i båda webbläsarna. Dessa två webbläsare är två av de största webbläsarna som finns men det finns ett antal andra kända webbläsare som inte mättes i detta arbete. Detta kan utföras i framtida arbeten för att se om samma resultat sker på alla webbläsare.

I detta arbete jämfördes båda teknikerna med tillämpningen bokhandel. I framtida arbeten går det att vidareutveckla och testa en annan tillämpning. Skillnaden mellan tillämpningarna kan exempelvis vara att skapa webbsida bestående av elektronikprodukter. Denna webbsida kommer förmodligen innehålla fler bilder och även videoklipp. Med det kan en slutsats dras om vilken teknik som är bäst anpassad för en webbsida innehållandes fler rörliga moment som exempelvis videos.

I den vetenskapliga artikeln skriven av Cao et al.(2010) där en webbsida byggs med hjälp av Joomla beskriver författaren att i framtida arbeten går det optimera Joomla-webbsidan med nya tekniker som ständigt utvecklas. Detta överensstämmer även med Xing et al.(2019) där författaren skriver att det går att göra mer forskning i front-end ramverk för att kunna optimera användandet av ramverken. Detta kan även utföras som framtida arbeten i denna

studie. I ett långsiktigt perspektiv går det att göra om denna studie om några år när teknikerna har utvecklats och se om resultaten blir annorlunda.

Ett annat perspektiv skulle vara att testa mellan teknikerna fast med en bokningssida. Detta kan exempelvis vara ett flygbolag eller taxibolag. Mätningarna kan då räkna svarstiden på hur en bokning går till eller hur snabbt en användare kan göra en inloggning.

Detta arbete kan även användas som en utgångspunkt vid skapandet av en e-commerce webbsida för en bokhandel. Företaget i fråga kan exempelvis efterfråga en teknik som möjliggör en webbsida med snabba svarstider. I detta arbete presenteras det två tekniker som jämförs utifrån svarstider. Företaget kan då fortsätta på experimentet och testa ifall slutsatsen som är dragen i detta arbete stämmer överens med deras resultat. Sedan går det skapa fler testfall för att vara säker på datan och utesluta fler faktorer samt optimera teknikerna.

Referenser

- apacherifends.org 2021. *Apache friends* Tillgänglig på Internet:
<<https://www.apachefriends.org/index.html>> [Hämtad 24 April 2021]
- Cao, X. and Yu, W., 2010. *Using Content Management System Joomla! to Build a Website for Research Institute Needs*. 2010 International Conference on Management and Service Science
- CSVJSON.com. 2021. *CSV to JSON - CSVJSON*. Tillgänglig på Internet:
<<https://CSVJSON.com/CSV2JSON>> [Hämtad 24 April 2021]
- Docs.Joomla.org. 2021. *J3.x:Creating a simple module/Developing a Basic Module - Joomla! Documentation*. Tillgänglig på Internet:
<https://docs.Joomla.org/J3.x:Creating_a_simple_module/Developing_a_Basic_Module>
- Docs.Joomla.org. 2021. *J3.x:Getting Started with Templates - Joomla! Documentation*. Tillgänglig på Internet:
<https://docs.Joomla.org/J3.x:Getting_Started_with_Templates/en>
- Gackenheim, C. (2015). *Introduction to React*. Apress
- GitHub. 2021. *a18abuah/examensarbete*. Tillgänglig på Internet:
<<https://GitHub.com/a18abuah/examensarbete>> [Accessed 24 May 2021]
- Intal, G., Robielos, R. and Ortega, A., 2018. *Design of user driven facility management system for universities*. Proceedings of the 6th International Conference on Information and Education Technology
- Kiatruangkrai, P., Phusayangkul, P., Viniyakul, S., Prompoon, N. and Kanongchaiyos, P., 2010. *Design and Development of Real-Time Communication Content Management System for E-Commerce*. 2010 Second International Symposium on Data, Privacy, and E-Commerce
- Lanford, P. and Hübscher, R., 2004. *Trustworthiness in e-commerce. Proceedings of the 42nd annual Southeast regional conference on* - ACM-SE 42
- Liang-fu, J., Jing-liang, C. and Yong-qin, S., 2010. *Website Design for Book Logistics Based on E-commerce*. 2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering
- Mickens, J., 2010. *Silo: Exploiting JavaScript and DOM Storage for Faster Page Loads*
- Mirdha, A., Jain, A. and Shah, K., 2014. *Comparative analysis of open source content management systems*. 2014 IEEE International Conference on Computational Intelligence and Computing Research

- Nikulchev, E., Kolyasnikov, P., Ilin, D., Kasatonov, S., Biryukov, D. and Zakharov, I., 2018. *Selection of Architectural Concept and Development Technologies for the Implementation of a Web-Based Platform for Psychology Research. Advances in Intelligent Systems and Computing*, pp.672-685
- Lewis, C., 2021. *React without npm, Babel, or webpack*. [online] Medium. Tillgänglig på: <<https://medium.com/@chrislewisdev/React-without-npm-babel-or-webpack-1e9a6049714>>
- Lipsum.com. 2021. *Lorem Ipsum - All the facts - Lipsum generator*. Tillgänglig på Internet: <<https://www.lipsum.com/feed/html>> [Hämtad 16 May 2021]
- Pano, A., Graziotin, D. and Abrahamsson, P., 2018. *Factors and actors leading to the adoption of a JavaScript framework. Empirical Software Engineering*, 23(6), pp.3503-3534.
- Patel, S., Rathod, V. and Parikh, S., 2011. *Joomla, Drupal and WordPress - a statistical comparison of open source CMS*. 3rd International Conference on Trendz in Information Sciences & Computing (TISC2011)
- Patel, S., Rathod, V. and Prajapati, J., 2013. *Comparative analysis of web security in open source content management system*. 2013 International Conference on Intelligent Systems and Signal Processing (ISSP)
- Reactjs.org. 2021. *Getting Started – React*. Tillgänglig på Internet: <<https://Reactjs.org/docs/getting-started.html>> [Hämtad 24 April 2021]
- Sharma, D. and Components, L., 2021. *Load and Render JSON Data into React Components* | Pluralsight. [online] Pluralsight.com. Tillgänglig på: <<https://www.pluralsight.com/guides/load-and-render-JSON-data-into-React-components>>
- StackOverflow. 2021. *How to implement navbar using React*. Tillgänglig på Internet: <<https://stackoverflow.com/questions/50166035/how-to-implement-navbar-using-React>> [Hämtad 24 April 2021]
- Tampermonkey.net. 2021. *Tampermonkey for Chrome*. Tillgänglig på Internet: <<https://www.Tampermonkey.net/>> [Hämtad 24 April 2021]
- Tomisa, M., Milkovic, M. and Cacic, M., 2019. *Performance Evaluation of Dynamic and Static WordPress-based Websites*. 2019 23rd International Computer Science and Engineering Conference (ICSEC)
- Voutilainen, J., Salonen, J. and Mikkonen, T., 2015. *On the Design of a Responsive User Interface for a Multi-device Web Service*. 2015 2nd ACM International Conference on Mobile Software Engineering and Systems

- Jiang W, Zhang M, Zhou B, Jiang Y and Zhang Y, 2014. *Responsive web design mode and application*. 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)
- J2Store - Joomla shopping cart. 2021. *Product Display Module*. Tillgänglig på Internet: <<https://www.j2store.org/extensions/modules/product-display-module.html>> [Hämtad 24 April 2021]
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B. and Wesslén, A., 2012. *Experimentation in Software Engineering*
- Xing, Y., Huang, J. and Lai, Y., 2019. *Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development*. Proceedings of the 2019 11th International Conference on Computer and Automation Engineering - ICCAE 2019