Examensarbete, Högskolan på Åland, Utbildningsprogrammet för Informationsteknik

WEBBAPPLIKATION BASERAD PÅ RAMVERKET REACT

- IMPLEMENTERING AV SOCIAL MEDIEPLATTFORM

Mathias Sjöblom

2023:39

Datum för godkännande: 19.12.2023
Handledare: Björn-Erik Zetterman

EXAMENSARBETE
Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Mathias Sjöblom
Arbetets namn:	Webbapplikation baserad på ramverket React - implementering av social medieplattform
Handledare:	Björn-Erik Zetterman
Uppdragsgivare:	

Abstrakt

Detta examensarbete beskriver hur jag med Javascript, React, Node.js och MongoDB skapade en social medieplattform. Syftet med detta är att lära mig i huvudsak frontend men även lite databashantering samt backend för att kunna ha som framtida referenser. Resultatet är en webbapplikation som uppfyller en stor del av mina kravspecifikationer men viktigaste av allt, att lära mig användbar teknik och teori för mitt framtida arbetsliv då detta är teknik jag vill arbeta inom och med.

Nyckelord (sökord)

Utveckling av webbapplikation, social medier, React, Javascript

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
2023:39	1458-1531	Svenska	42

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
20.09.2023	29.09.2023	19.12.2023

DEGREE THESIS
Åland University of Applied Sciences

Degree Programme:	Bachelor of Information Technology/Bachelor of Engineering
Author:	Mathias Sjöblom
Title:	Web application based on framework React - social media platform implementation
Academic Supervisor:	Björn-Erik Zetterman
Commissioned by:	

Abstract

This thesis describes how I created a social media platform using JavaScript, React, Node.js, and MongoDB. The purpose of this project is primarily to learn frontend development, along with some database management and backend aspects, for future reference. The result is a web application that meets a significant portion of my requirements, but most importantly, it serves as a valuable learning experience in practical technology and theory for my future career, as this is the technology I aspire to work with.

Keywords

Web application development, social media, React, Javascript

Serial number:	ISSN:	Language:	Number of pages:
2023:39	1458-1531	Swedish	42

Handed in:	Date of presentation:	Approved:
20.09.2023	29.09.2023	19.12.2023

INNEHÅLLSFÖRTECKNING

1. INLEDNING

I detta kapitel introducerar jag läsaren till arbetet och vad som behandlas inom det.

1.1 Syfte

Syftet med denna uppsats och projekt är att göra och dokumentera min process bakom att göra ett projekt som jag länge haft intresse att testa på att göra. Jag ska utveckla en social medieplattform som en webbapplikation där en användare kan logga in, göra inlägg, se andras inlägg, följa andra användare och lite till. Jag kommer att använda mig av några av de mest populära verktygen idag inom webbutveckling för att få en mer verklig och djupare insikt i dessa verktyg då det finns en framtid när jag använder dem i en professionell miljö. Jag gör även en liten forskningsdel i denna uppsats bland andra sociala medier för att se vad de använder sig av för att utmärka sig i dagens konkurrenskraftiga onlinemiljö och som jag möjligtvis kan använda mig av i mitt projekt. Jag hoppas på att lära mig:

- Frontend
    - UI design
    - UX
- MongoDB
- React
- NodeJS
- Egen tidshantering och projekthantering

1.2 Avgränsning

Inom ramen för detta arbete är min primära målsättning att uppfylla de specificerade skallkraven i kapitel 4. Allt som ligger utanför dessa krav betraktas som lyxfunktioner och jag planerar att implementera dem om jag har tillräcklig tid och kompetens utan att kompromissa med uppfyllandet av mina huvudmål. Detta fokus på att prioritera kärnkraven kommer att säkerställa att jag kan leverera en fungerande webbapplikation inom de angivna

ramarna och samtidigt skapa möjligheter för framtida förbättringar och utökningar. Denna utveckling kommer även endast vara optimerad för datorer. En optimering för mobila enheter kan vara en framtida förbättring.

1.3 Metod

Denna utveckling kommer att använda sig av ReactJS för frontend för att säkerhetsställa ett dynamiskt och responsivt användargränssnitt, NodeJS för att hantera användarautentisering, API endpoints och data routing med Postman för att skapa tester att dessa routes fungerar. MongoDB används för att hantera behandlingen och förvaring av användarna, inläggen m.m. Allt detta kommer att vara skrivet i Javascript vilket erbjuder enkel interaktion och utveckling av alla dessa tekniker tillsammans.

Visual Studio Code är mitt val av IDE, så det enkelt kan användas för alla tekniker och språk som jag kommer och kan tänka mig att använda under denna utveckling. All kod kommer att vara bevarad och versionshanterad på GitHub. En tidig design skapad i Photoshop kommer också att göras då jag vill ha en tidig förståelse för var mina komponenter på sidan skall placeras.

2. TEORI

I detta kapitel kommer jag gå igenom alla verktyg jag använde under min utveckling för detta projekt och förklara vad de innebär för att ge läsaren en bättre förståelse för framtida referenser i uppsatsen. Detta innehåller information om databaser, programmeringsspråk, ramverk för utveckling och lite programmeringsteknik som jag ville forska om.

2.1 MongoDB

MongoDB är en NoSQL-databas baserad på öppen källkod som såg dagens ljus år 2007 tack vare insatserna från Dwight Merriman, Eliot Horowitz och Kevin Ryan. Denna databashanterare har snabbt blivit ett kraftfullt verktyg för att lagra, hantera och hämta dokumentorienterad information, vilket gör den till ett värdefullt redskap för många applikationer och företag världen över.

En av de mest framträdande egenskaperna hos MongoDB är dess enastående flexibilitet och skalbarhet. Den är särskilt utformad för att hantera de moderna utmaningarna med att lagra och bearbeta enorma mängder data i realtid. MongoDB är känt för sin förmåga att hantera hög datavolym utan att tumma på prestanda och därmed möjliggöra skalning av system i takt med att behoven växer.

Det som särskiljer MongoDB från traditionella relationella databaser är dess struktur. Istället för den tabell- och radbaserade strukturen som används av SQL-databaser, använder MongoDB begreppet samlingar och dokument (collections and documents). Dokumenten är uppbyggda med hjälp av BSON (Binary JSON), en binär representation av JSON-data som ger en flexibel och snabb lagring av information. Denna struktur ger utvecklare och systemadministratörer friheten att anpassa sina databasstrukturer dynamiskt utan att behöva ändra scheman i förväg (*About Us - Our Story*, n.d.; Gillis & Botelho, 2023).

2.2 Javascript

Javascript är ett programmeringsspråk som erbjuder webbutvecklare att dynamiskt lägga till interaktioner till webbsidor, applikationer, servrar och även spel. I dagens digitala landskap är

Javascript det mest använda programmeringsspråket i världen och detta beror på flera goda skäl. En av dess framstående egenskaper är dess förmåga att samarbeta sömlöst med HTML och CSS. Detta innebär att det kan komplettera CSS genom att inte bara hantera utseendet och formateringen av HTML-element, utan också lägga till avancerade funktioner för användargränssnitt som öppnar dörren till en interaktiv och engagerande användarupplevelse, vilket är en förmåga CSS själv saknar och som har gett Javascript en central roll i utvecklingen av moderna webbapplikationer (Harder, 2018; Jordana, 2021). Här är några anledningar till varför Javascript är så populärt idag:

- Snabbhet - Det körs direkt med minimala laddningstider då det är kört i en webbläsare.
- Frontend och backend - Javascript kan användas för både frontend (react, angular, vue m.m) och för backend (node.js) vilket gör det enklare för frontend- och backendutvecklare att läsa, skriva och förstå varandras kod eller att göra båda delarna själv.
- Enkelt att använda - Javascript är relativt enkelt att förstå och komma igång med och det finns många bibliotek och tekniker för att göra mycket avancerade applikationer, så det är bra för nybörjare och avancerade användare (Brown, 2018; *Most Popular Programming Languages in 2023*, 2022).

2.3 HTML och CSS

HTML och CSS är två av de grundläggande byggstenarna för att skapa och forma moderna webbsidor där HTML definierar strukturen och innehållet och CSS styr utseendet och layouten.

HTML är en akronym för HyperText Markup Language som skapades år 1989 och var det första byggnadsblocket för att skapa en webbsida. HyperText beskriver HTML som "text wrapped within a text". Markup Language betyder att HTML är ett språk vars tillämpningsområde är formatering och layout av ett textdokument med dynamik och interaktiv text. HTML gör det möjligt att strukturera webbsidor med hjälp av olika taggar och element för att definiera rubriker, stycken, länkar, bilder och mycket mer (Fran, 2021).

CSS är en akronym för Cascading Style Sheet, vilket är ett enkelt designspråk som används för att göra processen att designa och styla en webbsida enklare. Med CSS kan du separera ett textdokuments innehåll från dess presentation. Det innebär att du kan ändra webbsidans utseende utan att ändra dess struktur. Genom att tillämpa olika stilar och regler på HTML-element kan du kontrollera färger, typsnitt, storlekar och layout. Detta gör det möjligt att skapa attraktiva och användarvänliga webbsidor med en konsekvent och professionell estetik (Gupta, 2020; *HTML Styles CSS*, n.d.).

2.4 Node.js

Node.js är en öppen källkod, plattformsoberoende körtidsmiljö som gör det möjligt att köra Javascript-kod. Vad som gör Node.js så kraftfullt och banbrytande är dess förmåga att göra det möjligt för utvecklare att skriva serverkod med JavaScript, samma språk som normalt används på klientsidan. Detta eliminerar behovet av att lära sig ett separat språk för server- och klientprogrammering och sparar därmed utvecklare betydande tid och resurser.

Node.js inkluderar JavaScripts V8-motor, samma motor som används av Google Chrome-webbläsaren för att tolka och köra kod. Detta innebär att Node.js kan kompilera JavaScript-kod internt med hjälp av JIT (Just-In-Time) kompileringprocesser som betyder att koden kompileras före körning. Resultatet är en imponerande ökning av hastigheten för körtiden, vilket möjliggör snabbare och mer effektiva serverapplikationer.

En av de mest revolutionerande aspekterna av Node.js är dess förmåga att hantera många samtidiga förfrågningar med en enda process. Till skillnad från traditionella serverprogram, som ofta måste skapa nya trådar för att hantera varje förfrågan, kan Node.js behandla tusentals förfrågningar samtidigt med en enda process. Detta eliminerar behovet av att oroa sig för antalet tillgängliga trådar och undviker de potentiella problem som är förknippade med multitrådning (Sheldon & Denman, 2022; *What Is node.Js? Complex Guide for 2023*, n.d.).

2.5 React

ReactJS är ett populärt Javascript-baserat UI-utvecklingsbiblioteket, vilket idag drivs av företaget Meta tillsammans med en grupp open-source-utvecklare och skapades av en Facebook-ingenjör år 2011 (Abba, 2022; Deshpande, 2020; Hutsulyak, 2023).

En av funktionerna som React erbjuder är koncepten "single-page application" och "reusable components". Med single-page application kan webbsidor laddas snabbt genom att enbart hämta de komponenter som behövs från React-biblioteket istället för att begära hela sidor från servern vid varje anrop. Detta resulterar i mycket snabba renderingstider och en övergång som känns nästan omedelbar för användaren.

Återanvändbara komponenter är en annan kärnfunktion i React. Denna förmåga gör det möjligt att skapa komponenter som är designade för att vara flexibla och modulära vilket gör dem enkla att återanvända på flera platser på en webbsida. Om du till exempel har en knappkomponent som du vill använda i olika delar av din webbapplikation, kan du enkelt implementera den igen och igen utan att skriva om koden. Detta främjar inte bara en enhetlig design över hela webbplatsen utan sparar också tid och minskar belastningen på servern (Abba, 2022; Deshpande, 2020; Hutsulyak, 2023).

2.6 SVG (Scalable Vector Graphics)

Scalable Vector Graphics (SVG) är ett filformat med en lång historia som sträcker sig tillbaka till 1990-talet, men det tog inte fart inom webbutvecklingen förrän år 2017. Detta filformat anses vara mycket användbart för webben, och istället för att lagra bilder som pixelbaserade filer, såsom PNG och JPG, använder SVG matematiska formler som bygger på linjer och punkter i ett rutnät.

SVG-filer är skrivna i XML och specificerar alla former, färger och text som utgör bilden. Detta möjliggör för utvecklare att manipulera SVG-filer med hjälp av egen Javascript- eller CSS-kod, vilket öppnar upp SVG-filernas fulla potential. En av de största fördelarna med

SVG är dess skalbarhet, eftersom vektorbaserade filer kan ändra storlek utan att förlora kvaliteten i den ursprungliga bilden. Detta är särskilt gynnsamt för projekt som inkluderar många ikoner och logotyper, där hög bildkvalitet är viktigt och behöver bibehållas oavsett storlek, vilket passar bra för mitt projekt. Med SVG kan du säkerställa att dina grafiska element ser skarpa och tydliga ut oavsett vilken enhet eller skärmstorlek som används. Detta gör SVG till ett utmärkt val för modern webbutveckling och design (*A Practical Guide*, n.d., *SVG Files: How to Create, Edit and Open Them*, n.d.; Juviler, 2022b).

2.7 Visual Studio Code

Visual Studio Code (VS Code) är en kraftfull kodredigerare utvecklad av Microsoft som släpptes år 2015 och har blivit ett av de mest populära verktygen i IT-världen. Den stöder en mängd olika programmeringsspråk och fungerar problemfritt i Windows, Linux och macOS, vilket ger utvecklare friheten att arbeta i sin föredragna miljö.

VS Code erbjuder ett stort bibliotek av användarskapade “extensions” som ger extra funktionalitet och anpassningsmöjligheter och som finns för att hjälpa med t.ex. felsökning, versionshantering, temahantering m.m. VS Code är också känt för sitt användarvänliga gränssnitt, intelligenta kodkomplettering och syntaxmarkering, vilket ökar produktiviteten och minskar risken för fel vilket sparar utvecklare en stor mängd tid och resurser.

I kombination med sin flexibilitet och mångsidighet har Visual Studio Code blivit en viktig del av många utvecklares arbetsflöden. Oavsett om du är nybörjare eller erfaren utvecklare så erbjuder VS Code de verktyg och funktioner du behöver för att göra kodning både effektiv och njutbar. Det är ett mångsidigt verktyg som gör det möjligt att snabbt och smidigt utveckla kvalitetsapplikationer (Heller, 2022).

2.8 Git och GitHub

Git är ett versionshanteringsverktyg skapat år 2005 av Linus Torvalds som tillåter utvecklare att spåra ändringar i sin kod. Med hjälp av “branching” och “merging” så görs det möjligt att arbeta parallellt utan att påverka andras kod. Spårningen av kod möjliggör det för utvecklare

att kunna gå till tidigare versioner av koden för att hantera fel när ändringar har gjorts (Wikipedia contributors, 2023a).

GitHub som skapades år 2008 bygger på Git är en webb- och molnbaserad service som erbjuder ett kraftfullt webbgränssnitt samt lagring som man kan använda för att enkelt se och följa sina "Git-repositories" som innehåller sin kod och dess tidigare versioner, vilket gör det möjligt att följa ändringar, hantera sina projekt och utvecklare att samarbeta på ett organiserat sätt (Lutkevich & Courtemanche, 2023; Wikipedia contributors, 2023d).

Git:s versionshantering möjliggör för utvecklare att effektivt följa och hantera ändringar i ett projekts kod på en stor skala. Två viktiga verktyg som bidrar till detta är branching och merging, och de spelar en avgörande roll i att göra arbetsflödet säkert och smidigt.

Branching är en funktion som tillåter en utvecklare att skapa en separat kopia (en branch) av projektets kod. Denna kopia fungerar som en isolerad arbetsplats där utvecklaren kan utföra ändringar, experimentera och utföra tester utan att riskera att påverka det huvudsakliga projektet eller andra som också arbetar på samma kodbas. Detta möjliggör parallellt arbete och möjliggör att flera utvecklare kan arbeta på olika funktioner eller buggfixar samtidigt utan att störa varandra.

Merging är processen där ändringar från en branch sammanfogas (merges) med den officiella koden i projektets huvudgren. När en utvecklare är nöjd med de ändringar de har gjort i sin branch och vill integrera dem med huvudprojektet, kan de använda merging för att sammanföra sina ändringar. Detta sker oftast efter att ändringarna har granskats och godkänts. Merging dokumenterar också de exakta ändringarna som har gjorts och inkluderar information om när och av vem ändringarna utfördes, så att det enkelt kan bli återställt ifall det behövs (Juviler, 2022a; *What Is GitHub? A Beginner's Introduction to GitHub*, 2018; Wikipedia contributors, 2023a).

2.9 Postman

Postman är ett utvecklingsverktyg som kan användas för att bygga, testa, dokumentera och modifiera API:er. Postman släpptes år 2012 och drivs idag av Postman Inc (Wikipedia contributors, 2023c).

En av Postmans mest framträdande funktioner är dess förmåga att underlätta skapandet av API:er. Med ett intuitivt användargränssnitt kan utvecklare enkelt skapa och definiera API-endpoints, inklusive HTTP-metoder, parametrar, autentisering och dataformat. Detta sparar tid och minimerar risken för felaktig konfiguration.

Postman är också till stor hjälp när det gäller att testa API:er. Utvecklare kan snabbt skapa och köra olika testfall för att verifiera att API:ets funktioner fungerar som förväntat. Testen kan anpassas för att täcka olika scenarier och resultatet presenteras på ett tydligt och överskådligt sätt.

För att underlätta samarbete och dokumentation erbjuder Postman möjligheten att dela och publicera API-dokumentation, vilket kan visa exempel på hur det används och mer detaljerad information om endpoints och parametrar.. Detta gör det enkelt att kommunicera och samarbeta med andra medlemmar i utvecklingsteamet och andra intressenter (*Introduction to Postman for API Development*, 2018, *Postman - API Testing Tool: What It Is, Tutorial - Javatpoint*, n.d., *What Is Postman?*, n.d.).

2.10 Photoshop

Photoshop är en rastergrafikredigerare utvecklad av Adobe för Windows och Mac. Photoshop kom ut på marknaden år 1990 för allmänheten och är idag ett av de mest kraftfulla programmen som används av alla möjliga branscher för grafisk design (Wikipedia contributors, 2023b).

Photoshop är en pixelredigerare, vilket betyder att programmet ändrar på själva pixlarna av bilden, vilket kan utnyttjas för att skapa till exempel logos, konst, modeller och porträtt. En

central del av Photoshop är funktionen av lager vilket gör det möjligt att organisera och strukturera ens arbete på ett logiskt sätt där varje lager är en transparent film som man kan lägga text, filter och annat på. Detta tillåter designers att enkelt hålla koll på alla separata delar i bilden och kunna göra ändringar på något utan att det påverkar de andra lagren.

Photoshop används även för fotoretuschering och bildmanipulation, vilket är mycket använt inom reklam, mode och mer för att skapa bilder som är estetiskt tilltalande. Det är mycket vanligt att till exempel ta bort objekt, rynkor, fläckar och andra oönskade delar (*Photoshop Basics: Understanding Layers*, n.d.; Ryan et al., n.d.; Smith, n.d.).

I mitt projekt användes Photoshop för att skapa en flödesdesign som jag kunde följa för att veta var på sidan jag ville att mina komponenter skulle vara, så att det var enklare att få positionerna rätt tidigt i projektet.

3. TIDIGARE FORSKNING

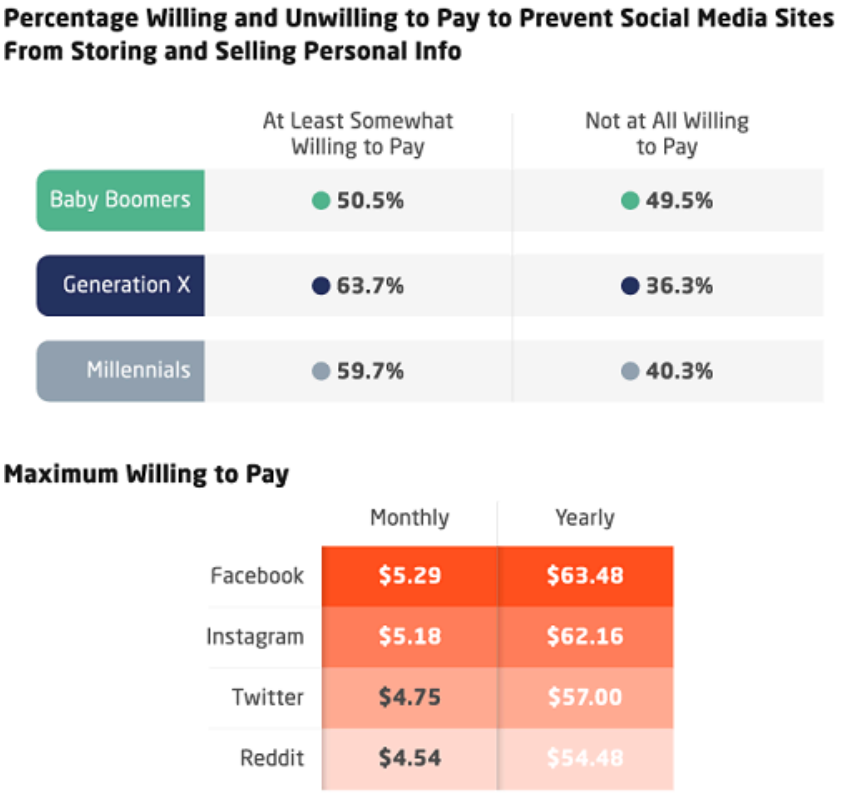
Jag ville även i denna uppsats göra en kort sammanfattning av tidigare forskning runt vad de populära sociala medierna har gjort för att lyckas i en så konkurrenskraftig verksamhet. Min målsättning är att se om det finns något jag kan lära mig och använda mig av i mitt projekt för att underlätta processen.

3.1 Varför är topplattformarna så populära?

Lättanvänt - En stor gemensam sak de populära plattformarna har är att de är designade på ett sätt så allt är lätt att komma åt och förstå även om man är obekant med internet vilket öppnar breddar plattformens åldersmålgrupp så även unga och äldre kan använda sig utav den (Belle Wong, 2023; Fita, 2012).

Lättillgängligt - Alla plattformar är enkla att komma åt och det spelar ingen roll vad för enhet man använder, det är snabbt och smidigt att använda plattformen antingen via en webbläsare eller via plattformens egen app (Belle Wong, 2023; Fita, 2012).

Kostnad - En stor orsak till en plattforms enorma population är att det är en gratis service. Det finns ingen stor social mediaplattform idag som tvingar sina användare att betala för att få använda den. När det finns så många kostnadsfria alternativ tillgängliga, varför investera pengar i något som är av lägre kvalitet än det som redan erbjuds gratis. En intressant undersökning som har gjorts om detta är att det finns en ganska stor folkgrupp som skulle vara villiga att betala för en sådan tjänst ifall det garanterar att deras personliga information är säkert förvarad och inte delas ut till ett tredje parti.



Figur 1. Undersökning från twingate.com gällande betalda social medier (Privacy for a Premium, n.d.)

Figur 1 visar en undersökning som gjordes år 2020 med 1 000 aktiva användare av social media. Vad denna undersökning berättar är att det faktiskt finns en grupp som är villiga att betala en månadskostnad för samma tjänst om skaparna bara gjorde plattformen säkrare för användaren. Denna undersökning skedde dock med endast amerikanska undersökande så ett annat resultat hade varit möjligt ifall demografin hade sett annorlunda ut. (Hutchinson, 2020).

Kortformat - Något de flesta sociala medier har i fokus är att hålla användarinnehållet kort och lätt konsumerat. Twitter har en maxgräns på 280 tecken per tweet (10 000 med twitter blue) för att hålla det snabbläst, Snapchat har också en maxgräns på hur långa videor man kan skicka (60 sekunder). Det är gjort så att man enkelt kan ta fram sin telefon och konsumera hur mycket tid som man har möjlighet till. Det spelar ingen roll om det bara är några minuter eller flera timmar som man kan spendera (*How People Become Popular on Twitter*, 2020; McCoy, 2023).

Anonymt - Jämfört med Facebook där man använder sitt riktiga namn och efternamn så använder man på Twitter, Snapchat m.m istället användarnamn där man kan använda vad som helst (inom ramarna för tjänstens användarvillkor) för att representera sig själv och denna representation behöver inte överhuvudtaget vara kopplad till din verkliga identitet. Detta erbjuder användarna att posta, kommentera, följa totalt anonymt vilket många tycker är en form av säkerhet och yttrandefrihet. Problemet med detta dock är att det kan leda till mer trakasserier, personliga attacker, hot m.m vilket leder till mer arbete för företagets support, men det är en uppoffring företagen gör (Dominic, 2021; Kuneff, 2021).

Fomo “Fear of missing out” - Fomo har stark psykologisk påverkan som har blivit en central drivkraft i moderna plattformar. Denna känsla av att missa något är något som används av alla typer av medier idag. Om du inte aktivt deltar på din personliga plattform under en vecka, kan du plötsligt känna att du har missat en hel del viktiga händelser och uppdateringar. Denna psykologiska faktor har blivit än mer utbredd i en tid där sociala medieplattformar och nyhetskanaler uppdateras i realtid. Människor vill hålla sig uppdaterade om vad deras vänner och bekanta gör, och de vill vara först med att fånga upp dagens heta nyheter. Fomo påverkar inte bara hur ofta vi använder våra digitala enheter utan också hur vi mäter vår egen självkänsla. Detta kan skapa en press att ständigt vara "uppkopplad" och "in-the-know" (*Social Media and FOMO*, 2022).

3.2 Varför misslyckades vissa stora plattformar?

Under internets korta existens så har många sociala plattformar kommit och gått. Mellan åren 2005 till 2008 var Myspace det mest populära nätverket på internetet med cirka 76 miljoner månatliga aktiva användare, medan Myspace idag endast har sex miljoner aktiva användare, vilket inte är en obetydlig mängds, men ingenting jämfört med de stora nätverken som har mellan hundratals miljoner till miljarder användare. Det är tänkt att anledningarna till att Myspace misslyckades att hålla sig i toppen var för dålig design, reklam överallt på sidan, dålig administration och att sidans publika image hade förfallit under åren vilket drev användare iväg från sidan. En viktig del i ett socialt nätverk är att det är många som använder

det, så använder ingen av dina kompisar eller bekanta nätverket är chansen att du kommer använda det väldigt låg (Sandu, 2023; The Editors of Encyclopedia Britannica, 2023; Zandt, 2021).

Figur 2. MySpace gamla logo

Ett intressant fall som skedde nyligen var Metas nya plattform Threads. Threads kom ut den 5 juli 2023 och nådde 100 miljoner användare efter endast fem dagar, jämfört med tre år för Snapchat och hela nio månader för TikTok. I augusti, månaden efter att Threads släpptes, rapporterades det att Threads hade endast 10,3 miljoner aktiva användare vilket är en 79 % nedgång från plattformens toppnotering. Det fanns många anledningar till varför Threads inte blev succén Meta hade hoppats på, till exempel dåligt med funktioner, tråkigt/dåligt innehåll, blev aldrig släppt i Europa med mera. Så om även Meta som är ett av de största företagen i världen kan misslyckas med att skapa ett nätverk är oddsen för andra som vill testa inte så bra för det visar hur mycket tid, pengar och resurser som krävs för att kunna pröva på denna verksamhet (Mc Gowran, 2023; Sainz, 2023).

Figur 3. Threads logo

4. KRAVSPECIFIKATION

Detta projekts största fokus är inom frontendutveckling. För att skapa en mer fullständig och komplett applikation som jag kan ha som framtida referens vill jag också ha ett par börkrav kopplade till backendutvecklingen, vilket även bidrar till en mer fullständig applikation i slutändan.

4.1 Skallkrav

Flera sidor/vyer skall finnas i systemet:

A. Inloggningssida, visas för alla
B. Registreringssida, visas för alla
C. Flödesvy, där användare ser inlägg från andra användare som användaren följer
D. Se sina egna inlägg och följare i sin profil
E. Använda interna API:er i frontend för att fylla med dynamisk data.
F. En fullständig design för alla sidor

4.2 Börkrav

API för intern interaktion mellan front-/backend, “backend for frontend"

G. Skapa textinlägg, möjligtvis bildinlägg
H. Gilla ett befintligt inlägg
I. Kommentera under ett befintligt inlägg
J. Följa/sluta följa en annan användare
K. Logga in / registrera ny användare

Databaskoppling från backend till MongoDB

A. Välja format för data (Schematics)
B. Skapande av tabeller
C. Koppla applikation med kod till databas
D. Skapa och läsa data

5. IMPLEMENTATION

I detta kapitel kommer jag skriva en fritt formulerad text samt visa work-in-progress bilder om min implementation av alla delar som behövdes för att uppfylla mina krav för en fullständig applikation som jag är nöjd med.

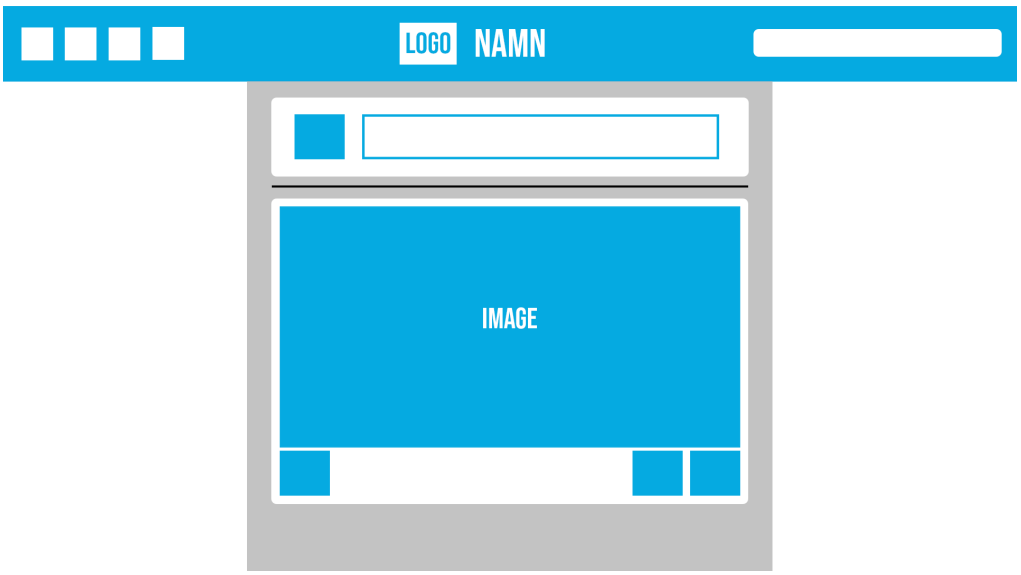
5.1 Planering

I början av projektet gick jag in lite halvblint och gjorde allting i den ordningen jag ville men märkte tidigt att det behövdes delas upp i delar för att veta vad som behövs göras när och var. Det här är ordningen jag bestämde var mest logisk att följa för att hjälpa mig uppnå så mycket önskat som möjligt.

1. Upplägg av sidorna (flöde, login, registrering, profil) med temporär design
2. Design av datamodell för MongoDB och Models
3. Implementera API (Skapa posts, skapa användare, gilla post, följa/sluta följa användare osv)
4. Koppling av API via klient-sidan
5. Slutgiltig design för alla sidor

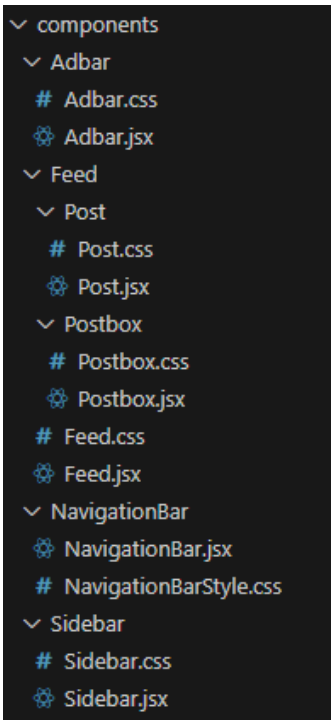
5.2 Upplägg av sidor och komponenter

Det första jag gjorde när jag började projektet var att skapa alla komponenter för själva flödessidan och placera dem enligt en design jag hade gjort i Photoshop för att veta ungefärligen hur och var jag ville ha komponenterna på sidan (se figur 4). Vid detta stadie i projektet visste jag inte att jag ville använda mig av SVGs så jag använde temporära bilder som jag lagrade i en mapp för publika resurser.



Figur 4. Första designen av flödesvyn gjord i Photoshop

Figuren nedan (figur 5) visar en lista på de komponenter jag gjorde för att lägga upp flödessidan. Den första komponenten är för en adbar vilket är den tomma ytan till höger i flödessidan. Anledningen till namnet är att jag tidigt i projektet hade en idé att testa att lägga in en Google Adsense där för att få lite erfarenhet inom det, men slutligen bestämde jag mig att överge den idén så komponenten är tom men har css för uppdelning av sidan så den är kvarstående för annars skulle sidans fördelning bli ojämn.



Figur 5. Komponentlista för flödessidan

Postkomponenten (se figur 6) är hur själva posten/inlägget en användare laddar upp är uppbyggd och ser ut. Komponenten innehåller följande:

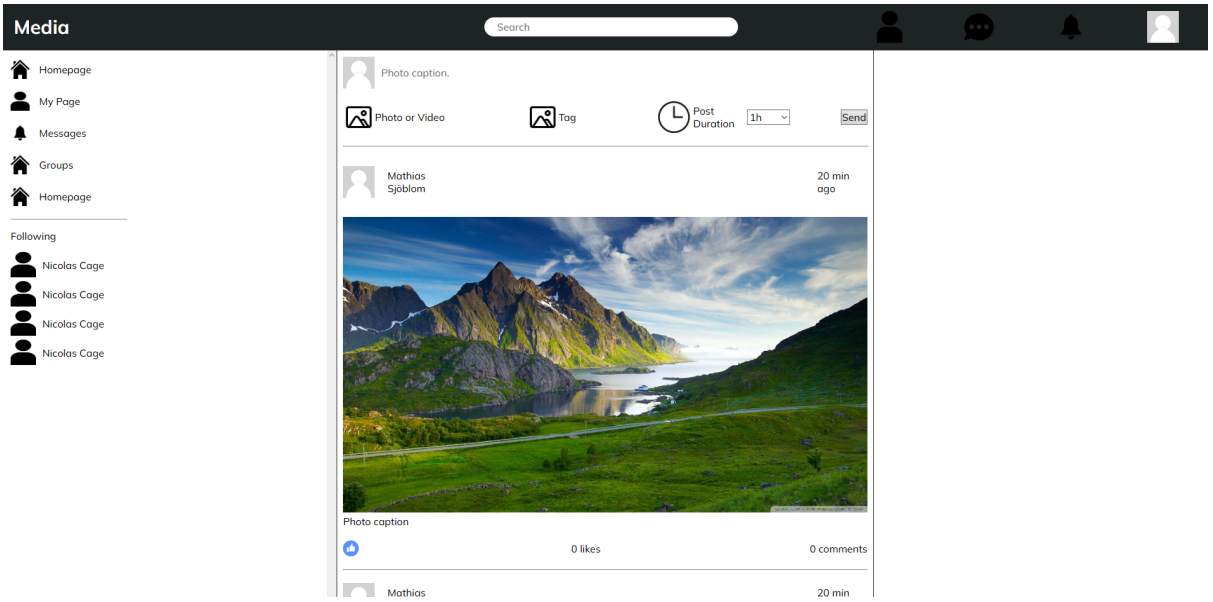
- Användarens avatar
- Användarens namn
- Datum vid uppladdning
- Uppladdad bild av användaren
- Text skriven av användaren
- Gilla knapp
- Text för antal gillningar
- Text för antal kommentarer
- Kommentarer av följare

Postboxkomponenten tillåter användaren att skriva text samt ladda upp en valfri bild i PNG- eller JPG-format. Detta sparas i en post i databasen ifall användaren klickar på att skicka inlägget. Personer som följer användaren kan se detta inlägg i deras feed. Inlägget kan även ses av personer som besöker användarens personliga profil där det finns en lista med användarens alla inlägg. Denna komponent används både i feeden samt i en personliga sida så man kan göra inlägg på två sidor, men de fungerar på samma sätt.

Feedkomponenten ansvarar för hur själva feeden ser ut och är uppställd med inlägg från andra användare du följer. Den innehåller API fetches för att hämta inlägg från användare du följer eller från en användare vars profil du är inne på. Denna komponent används två gånger, på huvudsidan samt på profilsidan.

Navigationbar-komponenten är delen som är högst upp på sidan där man kan se sidans logo som man kan klicka på för att ta användaren tillbaka till flödessidan. Komponenten innehåller en sökruta för att hitta användare samt bilder för att ta användaren till andra vyeri sidan t.ex. vänner, chatter, notifikationer osv men dessa är endast för utseende och har ingen riktig användning i denna del av projektet. Denna bar är “sticky” så även om man skrollar ner eller upp i sidan så kommer den alltid att vara högst upp för enkel tillgänglighet.

Sidebar-komponenten är den vänstra delen av sidan där man kan se några extra knappar samt en lista med personer som man följer som man kan enkelt klicka på för att ta sig till deras profil.



Figur 6. Första iterationen av flödesvyn

5.3 MongoDB-modeller och dokumentsamlingar

Ursprungligen hade jag planerat att använda mig av Firebase som databas men efter lite problem med att få tillgång till alla Firebase funktioner så bestämde jag mig för att ändra till MongoDB, vilket har fungerat perfekt för mitt projekt. Det är en NoSQL-databas, vilket betyder att den klarar av att hantera flexibla dokument istället för strikta tabellstrukturer. Det är särskilt användbart i miljöer där datamodellen kan förändras över tid utan omfattande omstrukturering, vilket ger projektet mer framtidssäkerhet. Jag började med att skapa ett User Schema vilket berättar till databasen hur jag vill att en användare ska se ut och innehålla (se figur 7). En användare i min databas ser ut som följande:

- Ett unikt användarnamn med minst tre tecken och max 15 tecken
- En unik e-post med minst 60 tecken
- Ett lösenord med minst 7 tecken
- En beskrivning med max 200 tecken som visas i användarens profil
- En avatar som används som profilbild
- En banner som kommer användas som en “bakgrundsbild” i profilen
- En lista med users som följer användaren

- En lista med users som användaren följer
- Ett booleskt värde för att se om användaren är en admin

```
const UserSchema = new mongoose.Schema({
    username: {
        required: true,
        unique: true,
        type: String,
        min: 3,
        max: 15
    },
    email:{
        required: true,
        unique: true,
        type: String,
        max: 60
    },
    password: {
        required: true,
        type: String,
        min: 7
    },
    description: {
        type: String,
        max: 200
    },
    avatar: {
        type: String,
        default:""
    },
    banner: {
        type: String,
        default:""
    },
    followers: {
        type: Array,
        default:[]
    },
    following: {
        type: Array,
        default:[]
    },
    admin: {
        type: Boolean,
        default:false
    }
},
{timestamps: true}
);
```

Figur 7. Usermodell

Efter userschemat så gjorde jag ett postschema för hur en användares inlägg ser ut och är uppbyggda, vilket är enligt följande:

- Ett userID för att visa vilken användare som gjorde posten
- En bild som användaren har laddat upp
- En lista med vilka personer som har gillat inlägget
- En beskrivning på max 100 tecken

```
const PostSchema = new mongoose.Schema({
    userId: {
        type: String,
        required: true
    },
    image: {
        type: String,
    },
    likes: {
        type: Array,
        default: []
    },
    description: {
        type: String,
        max: 100
    }
},
{timestamps: true}
);
```

Figur 8. Postmodell

Efter mina scheman så har jag två dokumentsamlingar i min databas, en som lagrar alla inlägg samt en som lagrar alla användare. Figuren nedanför (figur 9) visar hur objekten i tabellerna ser ut.

```
_id: ObjectId('640633895158d806d3bba8f2')
username: "Mathias"
email: "Mathias@gmail.com"
password: "Admin"
avatar: "Avatar.png"
banner: "Banner.png"
following: Array
followers: Array
admin: false
createdAt: 2023-03-06T18:40:09.625+00:00
updatedAt: 2023-03-06T19:40:24.914+00:00
__v: 0
```

```
_id: ObjectId('6406411e5a8b795c5d50a4ff')
userId: "640633895158d806d3bba8f2"
image: "test2.jpg"
likes: Array
description: "Mathias post"
createdAt: 2023-03-06T19:38:06.665+00:00
updatedAt: 2023-03-06T19:38:06.665+00:00
__v: 0
```

Figur 9. MongoDB-användare (vänster) inlägg av användare (höger)

5.4 API

Efter uppbyggnaden av databasen och scheman kunde jag börja med backend för frontend mönstret. Detta var delen jag fick mest problem med då det var länge sedan jag hade gjort något i Node.JS men som tur var fanns det mycket resurser online om just detta och så gick jag även en kurs i node, vilket gjorde allt lättare. En rutt (route) i ett API beskriver den specifika URL-sökvägen eller endpointen som används för att kommunicera med API:et. Varje rutt definierar hur en viss typ av request skall hanteras, till exempel skicka eller hämta

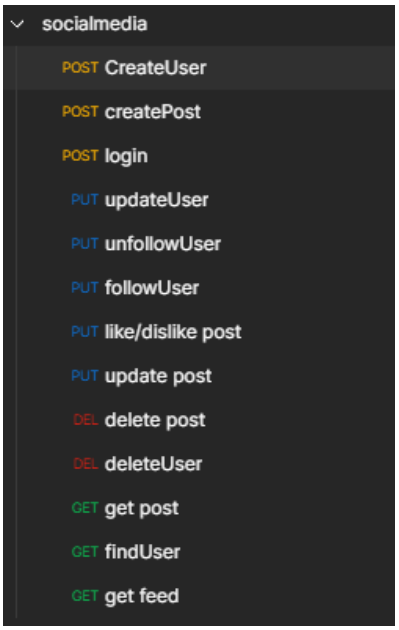
data av API:et. Alla rutter är ganska liknande varandra i kod så för att inte ha en mängd upprepade stycken kommer jag inte gå in i detalj i rutterna, i figur 10 kan man se rutten som hämtar en given användares inlägg, först hittar den användaren med det angivna användarnamnet och sedan hittar den alla inlägg från användaren med det användarnamnet. Dessa är rutterna jag behövde för att få all funktionalitet jag ville ha i backend samt framtida funktioner:

- Autentisering
    - Inloggning
    - Register
- Inlägg
    - Skapa inlägg
    - Uppdatera inlägg
    - Radera inlägg
    - Gilla/ogilla inlägg
    - Hämta inlägg
    - Hämta feed inlägg(posts av alla följare användaren följer)
    - Hämta specifik användares inlägg
- Användare
    - Uppdatera användare
    - Radera användare
    - Hitta användare
    - Följ användare
    - Avfölja användare

```
router.get("/profile/:username", async (request, response) => {
    try {
      const user = await User.findOne({ username: request.params.username });
      const posts = await Post.find({ userId: user._id });
      response.status(200).json(posts);
    } catch (response) {
      response.status(500).json(response);
    }
  });
```

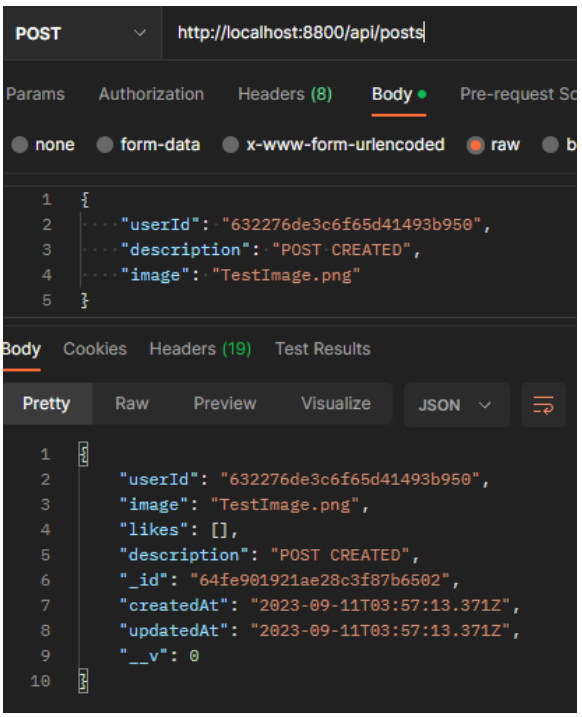
Figur 10. Exempel på en Node Route som hämtar en användares alla inlägg

Efter skapandet av varje rutt behövde jag även skapa ett test via Postman för att testa och se att datan kommer in i MongoDB korrekt och att felkontrollerna fungerar. Postman gjorde detta mycket enklare och smidigare då jag kunde göra förfrågningarna, felsökningarna och autentiseringen via ett lättanvänt gränssnitt. I figur 11 kan man se en lista över mina tester för funktioner så till exempel “CreateUser” testar att skapa en ny användare och lägger in användaren i databasen, detta gör det mycket enklare att se att själva funktionen fungerar utan att behöva lägga in temporär kod för att testa det vilket sparar utvecklare tid.



Figur 11. API test i Postman

Användargränssnittet i Postman gjorde det möjligt för mig att snabbt konfigurera och skicka förfrågningar till mina API-rutter. Jag kunde även utföra tester för att säkerställa att mina felkontroller fungerade som de skulle, vilket var viktigt för att skapa en robust och pålitlig backend. Figuren nedan (figur 12) visar hur jag enkelt via JSON-data kunde skapa ett inlägg som lades in i min databas.



Figur 12. Skapande av en post via Postman

5.5 Integration av API

Efter att API:et var uppbyggt var det äntligen dags att fylla på applikationen med funktionalitet och riktig data från MongoDB-databasen via API:et. Det första jag ville göra var att hämta inlägg från andra användare. Figuren nedan (figur 13) visar en bild på en UseEffect hook som kör ett API-anrop som hämtar en användares alla inlägg och lägger in dem i en tillståndsvariabel eller returnerar ett objekt som representerar en felsituation beroende på hur anropet gick. Efter det var det bara att mappa inläggets info till min postkomponent i flödet.

```
const [posts, setPosts] = useState([]);

useEffect(() => {
    const fetchPosts = async () => {
        try {
            const response = username ? await fetch("posts/profile/" + username) : await fetch("posts/feed/632276de3c6f65d41493b950");
            if (!response.ok) {
                throw new Error("Network response error");
            }
            const data = await response.json();
            setPosts(data);
            {/* console.log(data); */}
        } catch (error) {
            console.error("Error fetching data:", error);
        }
    }
    fetchPosts();
}, [username]);
```

Figur 13. Hämtande av post från användare

Sedan ville jag testa att hämta användaren av en profil och fylla profilen med data beroende på vilken användares profil det var. I figuren nedanför (figur 14) visas en kod för API-anropet som hämtar användaren beroende på vilket namn som ingår i den aktuella URL:en. UseParams-funktionen tillåter mig att hämta data som jag tidigare har namngett för enkel åtkomst, så om jag till exempel använder URL:en /profile/Mathias så hämtar den användarnamnet Mathias som jag sedan kan använda i mitt API-anrop för att hämta Mathias data för profilen.

```
const [user, setUser] = useState({});
const username = useParams().username;

useEffect(() => {
    const fetchUsers = async () => {
        try {
            const response = await fetch(`/users?username=${username}`);
            if (!response.ok) {
                throw new Error("Network response was not ok");
            }
            const data = await response.json();
            setUser(data);
        } catch (error) {
            console.error("Error fetching users:", error);
        }
    }
    fetchUsers();
  }, [username]);
```

Figur 14. Hämtande av användare från API

Loginsystemet var en annan klurig del att lista ut men som jag verkligen ville få fungerande. Jag fick använda mig av något jag inte har använt förut, vilket var ett context API. Som tur fanns det många videor och onlineresurser inom detta område så det gick bra att förstå allt som behövdes. Jag fick börja med att skapa en Context. Detta gör så att jag kan omsluta min App i en context provider, vilket gör att när en användare är inloggad så har jag tillgång till den inloggade användarens data från alla sidor som jag sedan kan använda för att personifiera sidorna med. I figur 15 kan man se hur “Context.Provider” omsluter "children", vilket är själva appen. Detta ger mig tillgång till context variabler som innehåller den inloggade användarens information i resten av projektet.

```
const INITIAL_STATE = {
    user: null,
    isFetching: false,
    error: false
};

export const Context = createContext(INITIAL_STATE);

export const ContextProvider = ({children}) => {
    const [state, dispatch] = useReducer(Reducer, INITIAL_STATE);
    return(
        <Context.Provider value={{user: state.user, isFetching: state.isFetching, error: state.error, dispatch,}}>
            {children}
        </Context.Provider>
    )
}
```

Figur 15. Contextfil

Figuren nedan (figur 16) visar en Actions-fil som håller koll på vilket stadie inloggningsprocessen befinner sig i. Det gör det möjligt för mig att använda en "type" för att indikera vilket stadie det är och vilken typ av data som skall skickas. Ifall en användare försöker logga in blir typen "LOGIN_START", ifall inloggningen lyckades så ändras typen till "LOGIN_DONE" och ett paket med användarens info skickas med, och ifall inloggningen misslyckades så ändras typen till "LOGIN_FAIL" och ett felobjekt skickas med i paketet.

```
export const LoginStart = () => ({
    type: "LOGIN_START",
});

export const LoginDone = (user) => ({
    type: "LOGIN_DONE",
    package: user,
});

export const LoginFail = (error) => ({
    type: "LOGIN_FAIL",
    package: error
});
```

Figur 16. Actions-fil

Nästa figur (figur 17) visar en Reducer vilket är en funktion som hanterar och modifierar tillståndet relaterat till inloggningen. Beroende vilket stadie som blev inskickad så får vi tillbaka data som är anpassad för det stadiet, så om logging har startat men inte nått slutet så blir caset "LOGIN_START" från typen deklarerad tidigare i actionfilen och ingen data ges. Om inloggningen lyckades blir caset "LOGIN_DONE" och användares info blir skickat i ett

paket och hämtningen (“fetching”) blir klar och avslutad genom att deklareras till falsk. Om inloggningen misslyckas så blir caset “LOGIN_FAIL” och ingen användarinfo ges, hämtningen stoppas och deklareras falsk och ett error blir skickat. Ifall inget av dessa cases blir skickade så får man en default respons vilket returnerar stadiet.

```
const Reducer = (state, action) => {
    switch(action.type) {
        case "LOGIN_START":
            return{
                user: null,
                isFetching: true,
                error: false,
            };
        case "LOGIN_DONE":
            return{
                user: action.package,
                isFetching: false,
                error: false,
            };
        case "LOGIN_FAIL":
            return{
                user: null,
                isFetching: false,
                error: true,
            };
        default:
            return state;
    }
}
```

Figur 17. Reducer-fil

Om typen är “LOGIN_DONE” så sker ett login API-anrop samt en node.JS route körs som kollar att en användare med den inskrivna emailen och lösenordet existerar i databasen som sker efter loginknappen har tryckts så tillåter den en användare som redan finns i MongoDB-databasen att logga in.

```
export const LoginApiCall = async (userInfo, dispatch) => {
    dispatch({ type: "LOGIN_START" });
    try {
      const response = await fetch("/auth/login", {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(userInfo),
      });
      const data = await response.json();
      if (response.ok) {
        dispatch({ type: "LOGIN_SUCCESS", package: data });
      } else {
        throw new Error(data);
      }
    } catch (error) {
      dispatch({ type: "LOGIN_FAIL", package: error });
    }
  };
```

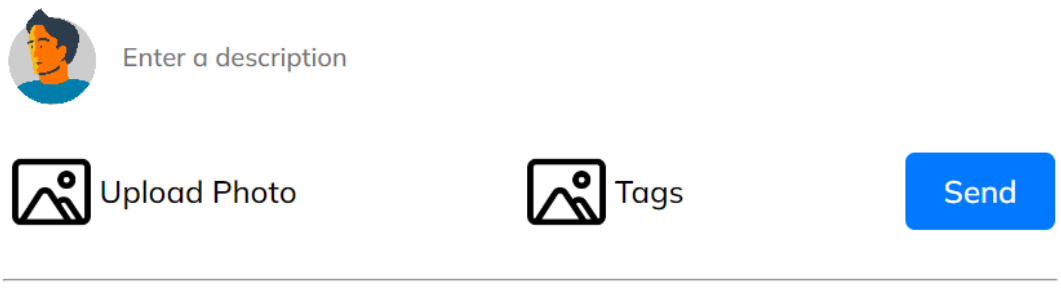
Figur 18. Login API-anrop

```
router.post("/login", async (request, response) => {
    try {
        const user = await User.findOne({email:request.body.email});
        !user && response.status(404).send("user error")

        if(request.body.password !== user.password) {
            response.status(400).json("password error");
        } else {
            response.status(200).json(user);
        }
    } catch (LoginError) {
        console.log(LoginError);
    }
})
```

Figur 19. Login API route

Att skapa ett inlägg är också en del som jag gärna vill ha fungerande. Högst upp i feeden och i profilen finns en ruta för användare att skapa ett inlägg. För användaren är det inte svårare än att skriva in en bit text, välja en bild och en tagg och trycka på “Send” med det sker lite mer i bakgrunden för utvecklarna.



Figur 20. Postbox för att skapa inlägg

Postrutan är uppbyggd så att det är ett formulär som tar in all data efter att “Send” -knappen har blivit tryckt. Formuläret har några krav, när man klickar på “Upload Photo” får man upp en ruta där man kan välja en bildfil från sin dator men endast de specificerade typerna av filer (png, jpeg, jpg), annars får man ett felmeddelande vid skickningen. När man trycker på Tags så får man upp en dialoglista där man kan välja en tag från en lista med hårdkodade ord. I detta skede av projektet kan man inte skicka en vald bild eller tagg men en del av backend:en finns där för framtida arbeten. Istället skickas texten man har skrivit in och en default bild från publicmappen.

Figuren nedan (figur 21) visar funktionen som aktiveras när skicka-knappen blir klickad efter att användaren har fyllt i allt. En ny post skapas som lägger in den inloggade användarens ID för att komma ihåg vems inlägg det är samt beskrivningen hämtas via en referens.

```
const {user} = useContext(Context);
const desc = useRef();

const PostHandler = async (e) => {
    e.preventDefault();
    const userPost = {
        userId: user._id,
        desc: desc.current.value
    }
    try {
        await fetch("/posts", userPost);
        window.location.reload();
    }catch(error) {
        console.error("Error creating post:", error);
    }
}
```

Figur 21. Funktion för att skicka inlägg

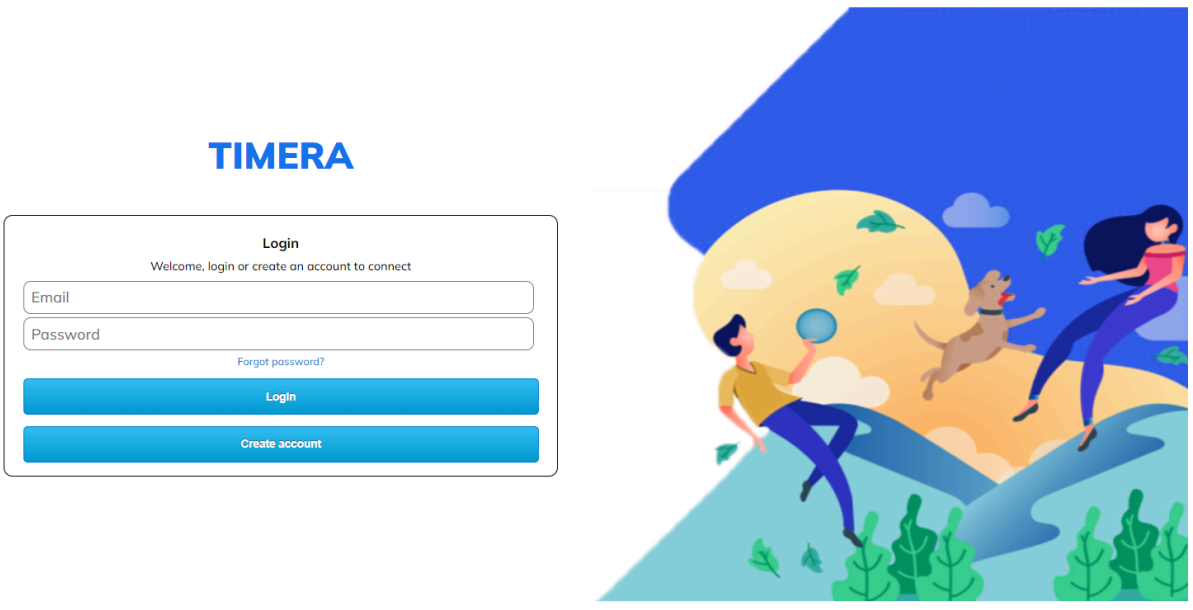
Efter att ett nytt inlägg har skapats med den skickade informationen i sig så görs en hämtning från API:et där datan tas emot och ifall allt gick bra sparas inlägget och en ett statusmeddelande HTTP 200 (OK) returneras. Vid en felsituation får man tillbaka ett statusmeddelande HTTP 500 (Internal Server Error). Nu kan inlägget ses av användare som följer användaren som skapade inlägget.

```
router.post("/", async (request, response) => {
    const newPost = new Post(request.body);
    try{
        const savedPost = await newPost.save();
        response.status(200).json(savedPost);
    } catch(err) {
        response.status(500).json(err);
    }
});
```

Figur 22. Node.js route för skapande av inlägg

5.6 Design och positionering

När jag var klar och nöjd med den funktionalitet som jag ville ha så kunde jag påbörja att designa sidorna som jag önskade. De sidor jag behövde göra en design till var inloggningssidan (figur 23), registreringssidan, flödesvyn (figur 24) och profilsidan (figur 26). Jag började med inloggnings- och registreringssidorna där jag skapade en enkel och lättförstådd design då detta är en av de viktigaste sidorna på grund av det är sidan som en ny användare först anländer till. Jag kom på ett enkelt temporärt namn för plattformen, "Timera", samt en enkel design och bild för att inte överkomplicera processen. Klickar man på "create account" -knappen så tas man till en liknande sida med annan text.



Figur 23. Inloggningssida

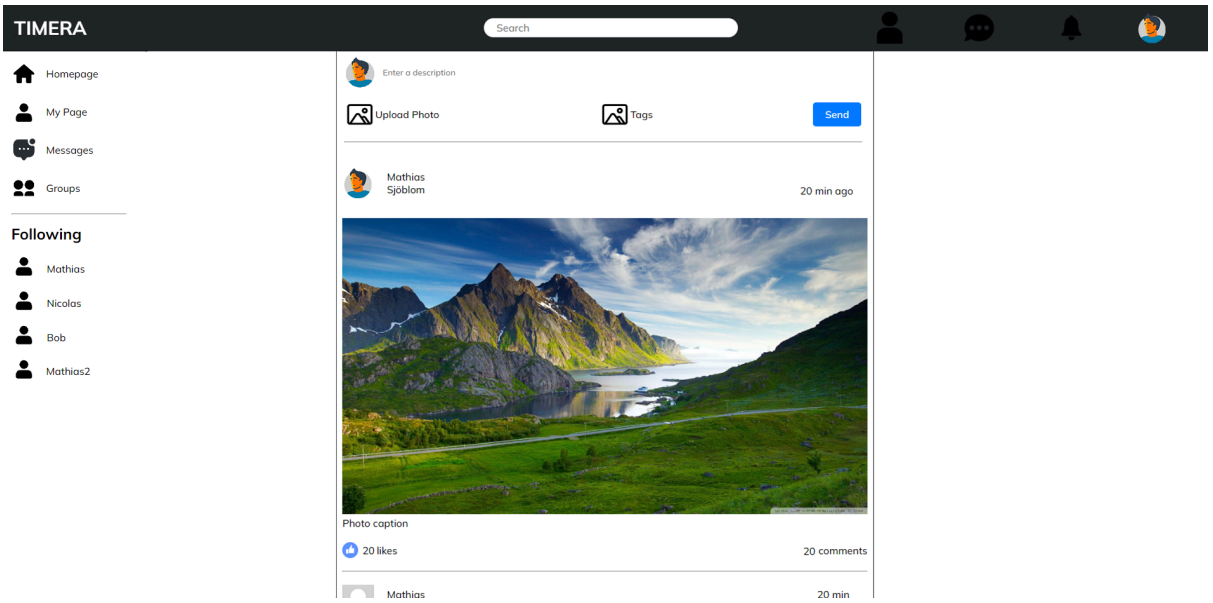
Det största projektet var att få en snygg flödessida (figur 24) där man enkelt förstår vad man ser och vad som händer. På toppen av sidan finns en navigeringskomponent som följer användaren även om de skrollar ner för enkel tillgång till andra sidor. Ifall man klickar på loggan uppe till vänster så tas man tillbaka högst upp i flödesvyn. Sökrutan i mitten tillåter användaren att hitta användares profiler och knapparna till höger har ingen funktionalitet men om man klickar på sin egen avatar längst till höger tas man till den inloggade användarens profil.

Till vänster finns en rad med några knappar som också leder till flödesvyn och profilen. Under "Following" -tabben kan man enkelt se en lista med alla användare man följer och klickar man på en användare så tas man till deras profil.

I mitten av sidan finns två delar. Högst upp finns postboxen som tillåter en användare att skapa inlägg, och under det finns det man kan anse som huvuddelen av applikationen, vilket är flödet där man enkelt kan se en lista med andra användares inlägg som man följer. Dessa inlägg kan man gilla och kanske i framtiden lämna en kommentar under.

Jag ville ha en enkel design och ett färgschema som inte skär i ögonen när man tittar på det, så detta är en sida som användare kan spendera mycket tid på. Jag gick in för en enkel blå färg som är populär bland andra plattformar då det anses vara som en lugnande färg. En liten linje lades mellan varje inlägg för att enkelt visa var inlägg tar slut och börjar. Varje bild visar också hur länge det var sedan inlägget blev skapat. Detta sker genom ett bibliotek som heter timeago.JS vilket tar "CreatedAt" -elementet från inlägget och formaterar det till ett enkelt förstått format som visar t.ex. "20 min ago". Något jag bestämde mig för att göra om, var hur ikonerna på sidorna fungerade. Jag hade ursprungligen alla ikoner i "image" -format vilket fungerar bra men det finns en populär metod som jag ville testa vilket innebar att byta ut dem till SVG format som gör det enklare att skala bilderna utan att förlora bildkvalité. Jag skapade en ny komponent för varje ikon (figur 25), vilket innehåller deras storlek, färg, samt andra attribut och en path som förklarar hur ikonen är uppbyggd. Jag försökte göra några själv men blev inte så nöjd med resultatet så istället tog jag några färdiggjorda från en gratissida men vill definitivt testa på att göra något större med denna teknik i ett annat projekt. Det var roligt

att få testa och lära sig om detta eftersom det var något jag var intresserad av och ser framtida användningar utav det.



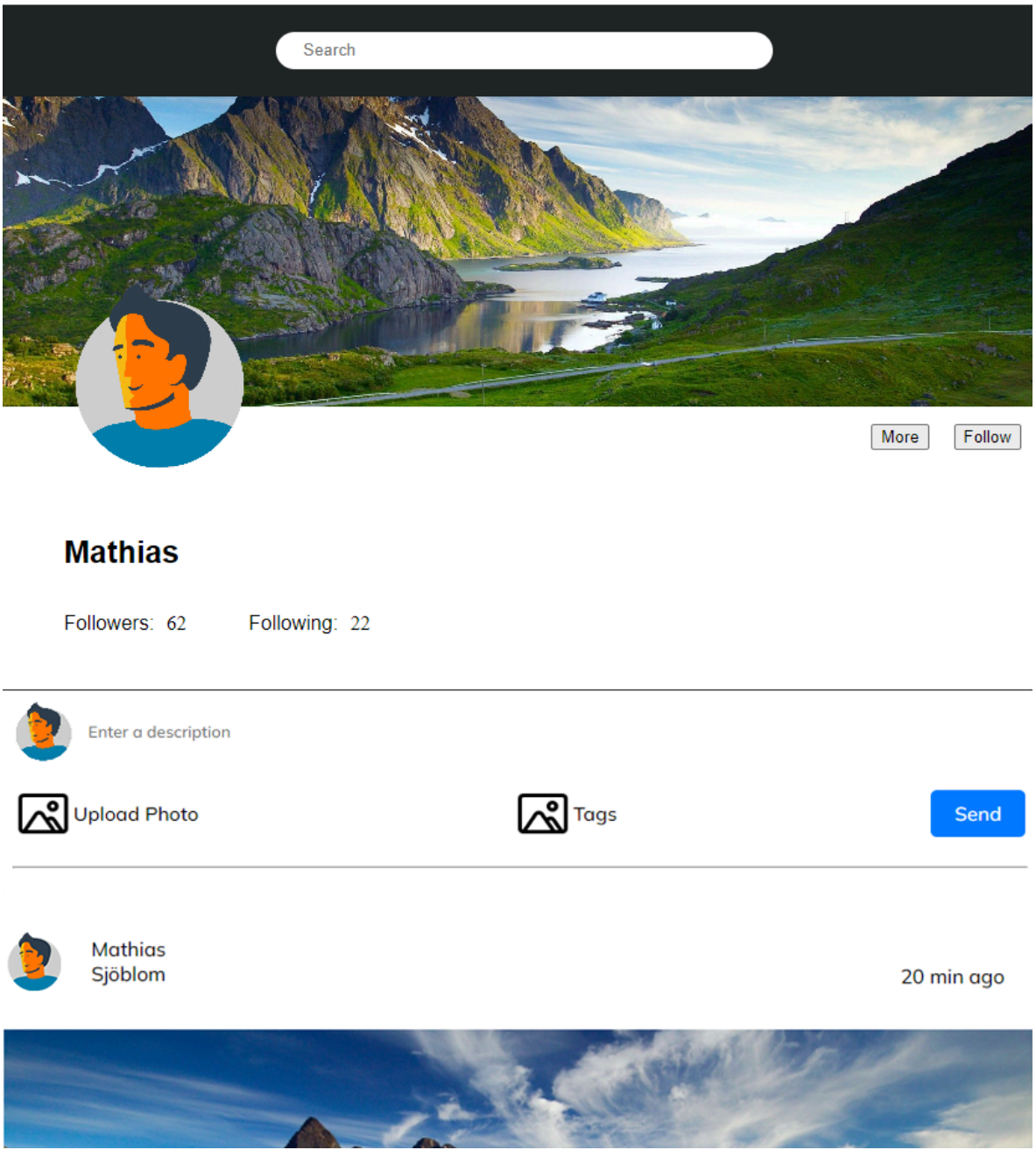
Figur 24. Flödessida

```
function SvgBell() {
    return (
        <svg fill="#000000" width="800px" height="800px" viewBox="-3 -2 24 24" preserveAspectRatio="xMinYMin" cursor="pointer" >
            <path d='M13.666 11.782L13 11.186V6a4 4 0 1 0-8 0v5.186l-.666.596A6.987 6.987 0 0 0 2.29 15h13.42a6.987 6.987 0 0
            0-2.044-3.218zM12 17a3 3 0 0 1-6 0H0a8.978 8.978 0 0 1 3-6.708V6a6 6 0 1 1 12
            0v4.292A8.978 8.978 0 0 1 18 17h-6zm-3 1a1 1 0 0 0 1-1H8a1 1 0 0 0 1 1z'/>
        </svg>
    );
}
```

Figur 25. SVG-komponent för notisikon

Den sista sidan är profilsidan, vilket också var en viktig del av projektet. På sidan (Se figur 26) ser man beroende på vilken användares sida det är information om just den användaren. Högst upp finns en bakgrundsbild och en avatar som användaren själv får bestämma men i detta skede används bara en bild i publicmappen som användaren har en sökväg till sparad i deras info för att göra det enklare att implementera i framtiden. Lite under bakgrundsbilden finns det knappar för att följa användaren vilket skapar ett API-anrop så att du kan se deras inlägg i flödet samt en “more” -knapp för framtida funktioner, t.ex. report, link profile osv. Under avatar finns en text som visar antalet följare användaren har samt en lista med antalet användare som användaren själv följer. Under allt detta så finns ett flöde med bara inlägg från användaren vars profil man besöker för enkel tillgång. Om man är inne på sin egen profil

finns postboxen ovanför flödet ifall man vill göra ett inlägg från profilen. Dessa inlägg fungerar på samma sätt som ett vanligt inlägg från flödesvyn.



Figur 26. Profilsida

6. SAMMANFATTNING och SLUTSATSER

Efter att ha genomfört denna uppsats och utveckling av min sociala medieplattform känner jag mig tillfredsställd och med ökat självförtroende inom utveckling. Genom att bygga och utveckla applikationen från grunden i mitt valda programmeringsspråk och ramverk har jag skaffat mig en betydligt ökad mängd kunskap och självsäkerhet inom ämnet och utveckling.

Specifikt var skapandet av API:et en utmaning som krävde extra ansträngning, men efter att ha överkommit den svårigheten känner jag nu ett betydligt ökad självförtroende inom detta område. Javascript, som var mitt val från början, fortsätter att vara mitt föredragna programmeringsspråk tack vare dess kraft och enkelhet. Valet att övergå från Firebase till MongoDB för databashantering visade sig vara givande och lättförståeligt. Även om jag är nöjd med detta beslut, ser jag fram emot att experimentera med Firebase i framtiden för framtida projekt.

Under projektets inledning hade jag ambitiösa mål (till exempel mobiloptimering och dark mode) men när jag insåg hur tidskrävande projektet var gav det mig en ökad förståelse för hur intensivt ett större projekt kan vara. Framöver, om jag skulle återgå till projektet, skulle jag noggrant titta vilka områden jag ville lära mig mer inom och använda mig av detta skal för att bygga ut och lära mig inom de områden där det är en bra grund för mycket av den teknik jag har intresse för.

Sammanfattningsvis är jag nöjd med de resultat jag har uppnått och ser fram emot att investera mer tid i projektet för att använda det som en referens och grund för vidareutveckling i framtiden. Detta projekt har inte bara utvecklat min tekniska kompetens utan även gett mig insikter om projektledning och planering för framtida professionella utmaningar.

KÄLLFÖRTECKNING

Abba, I. (2022, June 30). *What is react.Js? A look at the popular JavaScript library*. Kinsta. https://kinsta.com/knowledgebase/what-is-react-js/

About us - our story. (n.d.). MongoDB. Retrieved September 14, 2023, from https://www.mongodb.com/company

A practical guide. (n.d.). Retrieved September 14, 2023, from https://svgontheweb.com

Belle Wong, J. D. (2023, May 18). Top Social Media Statistics And Trends Of 2023. *Forbes Magazine*. https://www.forbes.com/advisor/business/social-media-statistics/

Brown, K. (2018, September 4). *JavaScript: How did it get so popular?* Codecademy Blog. https://www.codecademy.com/resources/blog/javascript-history-popularity/

Brown, S. (2020, December 4). *How people become popular on Twitter*. The Antelope Valley Times. https://theavtimes.com/2020/12/04/how-people-become-popular-on-twitter/

Deshpande, C. (2020, June 4). *The best guide to know what is React*. Simplilearn. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

Dominic, B. (2021, February 10). *Anonymity on social media and its ugly consequences*. Mint. https://www.livemint.com/opinion/columns/anonymity-on-social-media-and-its-ugly-consequences-11612977615856.html

Fita, M. (2012, November 28). *6 reasons why social networking is so popular these days*. Brandignity https://www.brandignity.com/2012/11/6-reasons-why-social-networking-is-so-popular-these-day/

Fran. (2021, June 3). *What are HTML and CSS used for?* FutureLearn. https://www.futurelearn.com/info/blog/what-are-html-css-basics-of-coding

Gillis, A. S., & Botelho, B. (2023, March 7). *MongoDB*. Data Management; TechTarget. https://www.techtarget.com/searchdatamanagement/definition/MongoDB

Gupta, A. (2020, October 9). *Difference between HTML and CSS: A complete guide*. Simplilearn. https://www.simplilearn.com/tutorials/html-tutorial/html-vs-css

Harder, J. (2018). What Is JavaScript? In *Graphics and Multimedia for the Web with Adobe Creative Cloud* (pp. 881–898). Apress.

Heller, M. (2022, July 8). *What is Visual Studio Code? Microsoft's extensible code editor*. InfoWorld. https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html

HTML styles CSS. (n.d.). W3Schools. Retrieved September 14, 2023, from https://www.w3schools.com/html/html_css.asp

Hutchinson, A. (2020, April 14). *Would people pay to use social media platforms to avoid data-sharing? [infographic]*. Social Media Today. https://www.socialmediatoday.com/news/would-people-pay-to-use-social-media-platforms-to-avoid-data-sharing-info/575956/

Hutsulyak, O. (2023, March 12). *Why use react for web development: 10 reasons to apply*. TechMagic. https://www.techmagic.co/blog/why-we-use-react-js-in-the-development/

Introduction to postman for API development. (2018, May 28). GeeksforGeeks. https://www.geeksforgeeks.org/introduction-postman-api-development/

Jordana, A. (2021, August 3). *What is JavaScript? A basic introduction to JS for beginners*. Hostinger Tutorials. https://www.hostinger.com/tutorials/what-is-javascript

Juviler, J. (2022a, October 31). *What is GitHub? (and what is it used for?)*. HubSpot. https://blog.hubspot.com/website/what-is-github-used-for

Juviler, J. (2022b, December 22). *SVG files: What they are and how to make one*. HubSpot. https://blog.hubspot.com/website/what-is-an-svg-file

Kuneff, N. (2021, November 3). *Advantages of anonymity in social media*. The West Boca Bullseye. https://wbhsbullseye.com/2929/news/advantages-of-anonymity-in-social-media/

Lutkevich, B., & Courtemanche, M. (2023, February 21). *GitHub*. IT Operations; TechTarget. https://www.techtarget.com/searchitoperations/definition/GitHub

McCoy, J. (2023, February 17). *What is short-form content? Your guide to when and how to use it*. Search Engine Land. https://searchengineland.com/short-form-content-393191

Mc Gowran, L. (2023, August 17). *From rise to fall: Where did Threads go wrong?* Silicon Republic. https://www.siliconrepublic.com/business/threads-where-did-it-go-wrong-twitter-x

Most popular programming languages in 2023. (2022, March 31). Coursera. https://www.coursera.org/articles/popular-programming-languages

Photoshop basics: Understanding layers. (n.d.). GCFGlobal.org. Retrieved September 14, 2023, from https://edu.gcfglobal.org/en/photoshopbasics/understanding-layers/1/

Postman - API Testing Tool: What It is, Tutorial - javatpoint. (n.d.). JavaTpoint. Retrieved September 14, 2023, from https://www.javatpoint.com/postman

Privacy for a premium. (n.d.). TwinGate. Retrieved December 12, 2023, from https://www.twingate.com/research/privacy-for-a-premium-exploring-peoples-sentiments-on-paying-for-social-media

Ryan, B., Tech, iD, & Moore, J. R. (n.d.). *What is Photoshop?* iD Tech. Retrieved September 14, 2023, from https://www.idtech.com/blog/what-is-photoshop

Sainz, R. R. (2023, July 23). *Threads fails, loses 70% of its active users*. Voz.us. https://voz.us/threads-fails-loses-70-of-its-active-users/?lang=en

Sandu, B. (2023, March 20). *Why did MySpace fail and how to avoid this in your Startup*. Upcut Studio. https://upcutstudio.com/why-did-myspace-fail/

Sheldon, R., & Denman, J. (2022, November 30). *Node.Js (node)*. TechTarget. https://www.techtarget.com/whatis/definition/Nodejs

Smith, J. (2022, Mars 18). *What is Photoshop*. American Graphics Institute. Retrieved September 14, 2023, from https://www.agitraining.com/adobe/photoshop/classes/what-is-photoshop

Social media and FOMO. (2022, October 17). Social Media Victims Law Center PLLC. https://socialmediavictims.org/mental-health/fomo/

SVG files: How to create, edit and open them. (n.d.). Adobe. Retrieved September 14, 2023, from

https://www.adobe.com/creativecloud/file-types/image/vector/svg-file.html

The Editors of Encyclopedia Britannica. (2023). Myspace. In *Encyclopedia Britannica*. https://www.britannica.com/topic/Myspace

What is GitHub? A beginner's introduction to GitHub. (2018, April 20). Kinsta. https://kinsta.com/knowledgebase/what-is-github/

What is node.Js? Complex guide for 2023. (n.d.). Netguru. Retrieved September 14, 2023, from https://www.netguru.com/glossary/node-js

What is postman? (n.d.). Postman API Platform. Retrieved September 14, 2023, from https://www.postman.com/product/what-is-postman/

Wikipedia contributors. (2023a, August 6). *Git*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Git&oldid=1169072219

Wikipedia contributors. (2023b, August 26). *Adobe Photoshop*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Adobe_Photoshop&oldid=1172339710

Wikipedia contributors. (2023c, September 9). *Postman (software)*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Postman_(software)&oldid=1174606652

Wikipedia contributors. (2023d, December 10). *GitHub*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=GitHub&oldid=1189218814

Zandt, F. (2021, November 12). *The rise and fall of MySpace*. Statista. https://www.statista.com/chart/26176/estimated-number-of-myspace-users-at-key-milestones/