



Chas Academy  
Fullstack .NET

# OptTheShop



**Examensarbete 2024**

Författare: Leo Stålenhag  
Handledare: Konrad Persson

# ABSTRACT

This report consists of my examination project, which is a full-stack web application aimed to optimize online shops on the platform Shopify. The application's frontend is developed in the JS library React, and the backend is developed in C#. By using ASP.NET and Microsoft Azure's cloud services to host the API, along with Blob Storage and KeyVault, I developed a specific tool for web developers and web shop owners to efficiently optimize their shop's media content. Content such as alt-texts and compressed images are the focus points of this application, based on identified user needs. The application strives to be a useful and easy tool to use regardless of previous experience. The project was reviewed in frequent iterations alongside co-workers at Semurai, who also participated in prototype-based user testing. The final application is thought to be a user-friendly alternative to existing tools on the Shopify platform today. Furthermore, the application was developed in such a way that further expansion to other platforms would be available.

# INNEHÅLLSFÖRTECKNING

<b>1. INLEDNING.....</b>	<b>4</b>
1.1 BAKGRUND.....	4
1.2 SYFTE.....	5
1.3 AVGRÄNSNING.....	5
<b>2. METOD OCH GENOMFÖRANDE.....</b>	<b>6</b>
2.1 Introduktion.....	6
2.2 Projektmetodik.....	6
2.3 Terminologi- och begreppslista.....	6
2.4 Utvecklingsprocess.....	7
2.4.1 Förundersökning och Verktysval.....	7
2.4.2 Frontend-utveckling.....	7
2.4.3 Backend-utveckling.....	7
2.4.4 Bildhantering och Lagring.....	8
2.4.5 Integrationer och API:er.....	9
2.5 Testning och Validering.....	9
2.6 Sammanfattning.....	10
<b>3. RESULTAT.....</b>	<b>11</b>
3.1 Alt-texter.....	11
3.2 Azure Staging.....	12
3.3 Gränssnitt.....	12
<b>4. DISKUSSION.....</b>	<b>15</b>
4.1 Projektets framgångar.....	15
4.2 Utmaningar och hinder.....	16
4.3 Förslag på vidareutveckling.....	17
4.4 Avslutning.....	17
<b>5. KÄLLFÖRTECKNING.....</b>	<b>18</b>
<b>BILAGOR.....</b>	<b>19</b>

# 1. INLEDNING

I dagens digitala landskap är det avgörande att webbplatser inte bara är visuellt tilltalande utan också är snabba och tillgängliga. Bilder spelar en stor roll i webbplatserns estetik och funktionalitet, men alldeles för stora filer kan påverka laddningstider och användarupplevelsen negativt om de inte är optimerade. För att förbättra tillgängligheten för synskadade användare, är alt-texter viktiga. Dessa texter är även viktiga för att förbättra SEO (sökmotoroptimering) (Moz, 2024).

För att möta dessa behov har jag utvecklat en fullstack-applikation som effektivt optimerar bilder och genererar alt-texter. Detta verktyg säkerställer att bilder är komprimerade utan att förlora kvalitet, vilket resulterar i snabbare laddningstider och mindre dataåtgång.

Dessutom genererar verktyget beskrivande alt-texter som förbättrar webbplatsens tillgänglighet och sökmotor ranking. Genom att använda detta verktyg kan webbplatsägare skapa en mer inkluderande och optimerad internetupplevelse för alla besökare. (TinyPNG, 2024; W3C, 2024).

## 1.1 BAKGRUND

Semurai, som idag är en digital marknadsföringsbyrå, använder ett flertal olika verktyg för optimering och effektivisering av arbetet inom webbutveckling. De har identifierat betydande brister i både resultat och användarupplevelse hos dessa verktyg. Med den ökande populariteten hos plattformar som Shopify och WordPress, har behovet av användarvänliga verktyg som levererar högkvalitativa resultat blivit allt mer påtagligt. Detta för att Shopify och WordPress möjliggör för individer utan tidigare erfarenhet av webbutveckling att enkelt skapa hemsidor. Jag upplevde att detta var ett intressant område att fokusera på inför mitt examensarbete, särskilt eftersom jag tycker att tillgänglighet är viktigt inom allmänna utrymmen såsom internet. (W3C, 2024)

De nuvarande verktygen uppvisar brister som inkluderar förvirrande användargränssnitt och resultat som inte når upp till önskad kvalitetsnivå. Exempelvis genererar verktygen för alt-texter ofta resultat som baseras på filnamn eller ger alltför kortfattade och generella beskrivningar av vad bilden faktiskt representerar (OpenAI, 2024). Det är viktigt att dessa alt-texter beskriver motiven på bilderna så tydligt som möjligt, så att den som är i behov av förklaringen kan förstå unika egenskaper hos bilden. Detta blir särskilt viktigt inom sammanhanget av onlinehandel, där potentiella kunder vill kunna jämföra olika produkter med varandra - alltså blir unika egenskaper det som driver kundens val av produkt (Moz, 2024).

Användargränssnittet är en särskilt kritisk aspekt där nuvarande verktyg brister. Plattformar som Shopify, som riktar sig till användare utan nödvändig erfarenhet, erbjuder ofta gränssnitt som är förvirrande och svåra att navigera (se bilaga 1). Användarna har svårt att

hitta och använda de specifika funktioner de behöver för att uppnå sina mål. (Shopify Polaris, 2024)

En annan brist som identifierades var att liknande verktyg inte finns som enskilda applikationer utan är inbäddade i större, mer omfattande verktyg. Genom att dela upp eftertraktade funktioner så blir det enklare att filtrera bort de funktioner som man inte är intresserad av att använda, samt att det blir mindre förvirrande att hålla koll på applikationernas respektive funktioner. Genom att utveckla verktyget som en enskild applikation blir det ytterligare ett bidrag till en mer tillgänglig arbetsmiljö inom webbutveckling. (Geeks for Geeks, 2024)

## **1.2 SYFTE**

Semurai strävar efter att utveckla ett verktyg som kan leva upp till behoven och åtgärda identifierade brister som nämnts ovan. Verktöget är tänkt att vara en förbättring av de interna processerna och ett steg mot att öka tillgången till effektiv digital marknadsföring och webbutveckling. Detta kan i sin tur bidra till ett mer tillgängligt internet som sparar på resurser och inkluderar fler användare.

Semurais ambition är därför att utveckla ett verktyg som inte bara är funktionellt och effektivt, utan även intuitivt och användarvänligt. Verktöget ska stödja användare på alla nivåer av teknisk erfarenhet och kunskap, vilket säkerställer att även nybörjare kan dra nytta av dess funktioner utan att känna sig överväldigade eller frustrerade. (Schwaber & Sutherland, 2024)

Genom att förbättra användargränssnittet och säkerställa att resultaten från verktyget är av hög kvalitet, skulle de befintliga bristerna kunna minskas. Detta kommer inte bara att gynna Semurais egna interna processer, utan även öppna upp tillgången till avancerade digitala verktyg. Förhoppningen är att detta kan leda till ett mer effektivt och tillgängligt internet, där fler användare kan skapa och underhålla högkvalitativa webbplatser utan att möta de hinder som nuvarande verktyg ofta innebär.

Det valda projektet var dessutom en utmaning för mig själv och min egen inläring, då projektet innebar många moment som jag inte var familjär med. Jag saknade en del erfarenhet kring vissa verktyg innan projektet startades, vilket var en av mina motivationer till att välja just detta område.

## **1.3 AVGRÄNSNING**

För att anpassa arbetet inom ramen för mitt examensarbete utvecklade jag appen specifikt för Shopify och använde uteslutande molnlösningar i Azure. Detta verktyg är för närvarande endast integrerat med Shopify, men API:et som ligger bakom verktyget är utformat för att kunna återanvändas och anpassas för utveckling mot andra plattformar eller program.

## 2. METOD OCH GENOMFÖRANDE

### 2.1 Introduktion

I detta kapitel beskrivs metoden och genomförandet av projektet, som innefattar utvecklingen av en fullstack-applikation för Shopify. Projektet omfattade frontend-utveckling med React och Shopifys Polaris-komponenter, samt backend-utveckling med ASP.NET och integration av externa API:er såsom OpenAI och TinyPNG. Azure användes för lagring och hantering av bilder samt deras KeyVault för hantering av API-nycklar.

### 2.2 Projektmetodik

Projektet genomfördes enligt Scrum-metoden, vilket innebar att arbetet delades upp i två veckors-sprintar och dagliga stand-up-möten. Detta tillvägagångssätt möjliggjorde kontinuerlig leverans och utvärdering av framsteg, och regelbunden identifiering och åtgärdande av eventuella hinder.

### 2.3 Terminologi- och begreppslista

För att uppnå projektets mål och säkerställa en effektiv utvecklingsprocess, användes följande verktyg och teknologier:

**React:** React är ett JavaScript-bibliotek för att bygga användargränssnitt. Det underlättar skapandet av interaktiva och dynamiska webbapplikationer genom att låta utvecklare bygga UI-komponenter. Reacts komponentbaserad arkitektur gör att koden är modulär och återanvändbar, vilket bidrar till effektiv utveckling och underhåll av applikationen.

**ASP.NET:** ASP.NET är ett kraftfullt ramverk för webbutveckling tillhandahållet av Microsoft. Det är en del av .NET-plattformen och används för att bygga dynamiska webbplatser, webbapplikationer och webbtjänster. Med stöd för flera programmeringsspråk, inklusive C#, erbjuder ASP.NET verktyg och bibliotek som underlättar utveckling av skalbara och högpresterande applikationer.

**OpenAI API:** Detta API är ett sätt att kunna använda ChatGPT i egna applikationer eller program.

**TinyPNG API:** TinyPNG är en webbplats där man kan ladda upp bilder för komprimering och konvertering. De erbjuder också ett API för utvecklare. API:et kan även konvertera om bilder till olika format. (TinyPNG, 2024)

**Azure:** Azure är en molnplattform från Microsoft som erbjuder en mängd tjänster för att bygga, distribuera och hantera applikationer. Användningen av Azure kan säkerställa en

stabil och säker drift av API:et, samt ge möjlighet till att dra nytta av plattformens skalbarhet och säkerhetsfunktioner. (Microsoft, 2024)

**Gadget:** En serverlös fullstack-plattform som specialiserar sig på att underlätta skapandet av Shopify-appar. Gadget hanterar hela integrationsprocessen med Shopify, inklusive autentisering, integration med Shopify-butiken, skapande av modeller baserade på filer och produkter i Shopify, samt automatisk generering av ett API för Shopify. (Gadget, 2024)

## 2.4 Utvecklingsprocess

### 2.4.1 Förundersökning och Verktysval

Innan skapandet av verktyget genomfördes en förundersökning för att identifiera de bästa verktygen och plattformarna för projektet. Under denna undersökning upptäcktes Gadget, vilket valdes på grund av dess förmåga att snabbt skapa en stabil bas för projektet och underlätta utvecklingen av Shopify-appar. Gadget valdes som utvecklingsplattform då de har väletablerade plugins, såsom deras Shopify-plugin, vilket snabbt gav insikt i dataflöden och användning av modeller med väldokumenterade och autogenererade API:er baserade på hur modellerna är uppbyggda.

### 2.4.2 Frontend-utveckling

**Verktyg:** React och Polaris

För att säkerställa ett enhetligt användargränssnitt använde jag mig av React för frontend-utveckling och Shopifys Polaris-bibliotek. Polaris-biblioteket innehåller en mängd React-komponenter som tillåter utvecklare att skapa gränssnitt som presenterar data på ett enhetligt och estetiskt tilltalande sätt. Detta eftersom Polaris följer Shopifys designprinciper, vilket underlättar skapandet av konsekventa och användarvänliga gränssnitt. React används av Gadget då allt som kan utvecklas med JavaScript kan utvecklas i Gadget. (React, 2024)

En annan anledning till att dessa bibliotek fungerar effektivt är att den modulära strukturen i React gör det enkelt att skala och anpassa applikationen vid behov. Detta öppnar upp för utvidgning av applikationen vid senare tillfälle.

### 2.4.3 Backend-utveckling

**Verktyg:** ASP.NET

ASP.NET användes för utveckling av API:et för att hantera kommunikation mellan frontend-delen och externa tjänster som OpenAI och TinyPNG. Syftet med att bygga detta API var att skapa en säker och effektiv plattform för dataflöde, vilket möjliggör enkel och konsekvent hantering av information. Dessutom bidrar centraliseringen av kommunikationen i ett API till förbättrad säkerhet genom kontroll över dataflöde och bättre

autentisering. Trots att Node.js är ett populärt val och används av Gadget, valde jag att utveckla ett externt API med ASP.NET för att möjliggöra återanvändning i andra applikationer och för att uppnå en SoC(*Separation of Concerns*). (Geeks for Geeks, 2024) Denna separation av frontend och backend gör applikationens system mer modulärt, vilket underlättar underhåll och framtida uppdateringar.

### Hosting på Azure

Genom att hosta API:t på Azure kunde jag samla all kommunikation till externa tjänster samt bildlagring på ett och samma ställe. Azure användes även för tillfällig lagring av komprimerade bilder och för att säkerställa säker förvaring av API-nycklar med Azure KeyVault. Jag kommer att förklara närmare hur jag använde Azure KeyVault i nedanstående avsnitt. Azure AppService användes för att hosta det API som byggdes för att lättare kunna hantera data från applikation enligt företagets riktlinjer. (Microsoft, 2024)

Genom att använda ASP.NET och Azure kan man skapa en robust och flexibel infrastruktur för applikationens kommunikation med externa tjänster. Detta kan bidra till en effektivare hantering av resurser.

#### 2.4.4 Bildhantering och Lagring

**Verktyg:** Azure Blob Storage och Azure KeyVault

Azure Blob Storage är en molnbaserad datalagringslösning som användes tidigt i projektet för att spara bilder under tester och utveckling. Denna tjänst erbjuder skalbar lagring för stora mängder ostrukturerad data, såsom bilder och videor, vilket underlättar utveckling och testning. I början av projektet sparades bilderna i Azure Blob Storage för att kunna testas mot OpenAI och TinyPNG API:er. Detta steg var avgörande för att säkerställa att bilderna inte blev korrupta och att API:erna gav korrekta och användbara responser.

För att använda OpenAI och TinyPNG krävs API-nycklar, vilket innebär en säkerhetsrisk om dessa nycklar inte hanteras korrekt. För att skydda API-nycklarna använde jag Azure Key Vault, en tjänst som hjälper till att säkert hantera och kontrollera åtkomsten till hemligheter som API-nycklar, lösenord och certifikat. Detta kunde säkerställa att nycklarna inte läckte ut och att endast behöriga delar av applikationen hade tillgång till dem. (Microsoft, 2024)

### Staging

En "staging URL" är en tillfällig webbadress som genereras för en resurs, såsom en bild, som är lagrad i ett mellanlagring område (Milecia, 2019). I detta sammanhang används Azure Blob Storage för att temporärt lagra bilder och skapa en tillfällig URL. Denna URL är sedan tänkt att skicka bilden till en annan tjänst, i detta fall Shopify, för permanent hosting. (Microsoft, 2024) Att använda sig av staging kan bli kostnadseffektivt för att lagringskostnaderna hålls nere, eftersom att ägandet av den långsiktiga lagringen flyttas till Shopify. En annan fördel av att använda staging i detta sammanhang är regelefterlevnad och säkerhet, eftersom tanken är att endast godkända bilder ska lagras långsiktigt.



## 2.4.5 Integrationer och API:er

**Verktyg:** OpenAI, TinyPNG, Shopify

OpenAI användes för att generera alt-texter baserat på bilddata. Genom OpenAIs API är tanken att applikationen tar emot metadata från bilder i frontend, bearbetar informationen och returnerar en alt-text. För att få fram ett bra resultat krävs ett stort fokus på de instruktioner som ges till OpenAI. OpenAI består av flera olika modeller med unika egenskaper. Jag använde mig av deras modell kallad "Vision" för att analysera bilderna och generera alt-texter. Dessutom använde jag deras textbaserade modell GPT-3.5 för att hantera textdata. Detta val gjordes för att spara på kostnader och optimera responsen från API:et.

Genom att använda dessa två modeller från OpenAI ville jag säkerställa att bilderna inte bara var välbeskrivna, utan också att deras filnamn och URL:er var optimerade för SEO, vilket innebär förbättrad synlighet och ranking i sökmotorer. Ytterligare har TinyPNG ett API för utvecklare, vilket jag använde för att komprimera bilder och för att kunna optimera laddningstider. Bilderna skickades från frontend till TinyPNG, där de komprimeras och sedan lagras temporärt i Azure Blob Storage innan de skickas tillbaka till Shopify för hosting.

Eftersom applikationen skulle utvecklas specifikt för Shopify, fanns det redan goda möjligheter till enkel autentisering och hantering av butiksdata. Integrationen av min applikation och Shopify hanteras av Gadget. Detta inkluderade skapandet av modeller baserade på Shopify-data och generering av API:er för butiksinteraktioner. Gadget underlättade utvecklingen genom att automatiskt hantera autentisering och dataflöden, vilket var tänkt att spara tid och resurser. Genom att använda Gadget var min förhoppning att jag skulle kunna bygga en robust applikation som var väl integrerad med Shopifys ekosystem.

## 2.5 Testning och Validering

**Verktyg:** Postman

För att säkerställa att applikationen fungerade korrekt valde jag att använda mig av integrationstester och användartester. Postman användes för omfattande API-testning för att säkerställa att alla API-anrop fungerade korrekt och att dataöverföringen mellan frontend och backend var pålitlig och effektiv. Detta innefattade testning av olika scenarier och belastningar för att identifiera och åtgärda eventuella problem tidigt i utvecklingsprocessen.

För användartester skapades prototyper i React, vilket gjorde det möjligt att snabbt iterera och förbättra gränssnittet baserat på feedback. Användarna som deltog i dessa tester bestod av medarbetare hos Semurai. En stor del av testerna fokuserade på användarupplevelse och användbarhet. Målet med användar testerna var att säkerställa att appen var intuitiv och lätt att använda, även för personer med begränsad teknisk kunskap. Hos Semurai finns det en

varierande kunskapsnivå hos medarbetarna när det gäller teknisk erfarenhet och kunskap. Därför kunde dessa användartester göras inom företaget utan att viss partiskhet skulle påverka resultatet.

## **2.6 Sammanfattning**

Tanken bakom detta urval av utvecklingsverktyg är bl.a. att hålla applikationen kostnadseffektiv, enkel att underhålla och uppdatera, samt att hålla den säker utan att behöva offra en god användarupplevelse. Den iterativa arbetsprocessen var till för att säkerställa kontinuerlig förbättring och anpassning till förändrade krav, vilket möjligtvis kunde resultera i en robust och användarvänlig applikation för Shopify.

Genom att följa Scrum-metoden kunde arbetet utvärderas iterativt under projektperioden. Förundersökningen ledde till att Gadget upptäcktes vilket gav projektet en stabil bas för utveckling. Projektets frontend utnyttjar en kombination av moderna utvecklingsverktyg och teknologier, såsom biblioteket React och Shopifys Polaris-komponenter. Tillsammans med en backend byggd kring ASP.NET och Azure har projektet en grund som är både expanderbar och modulär. Azure Blob Storage och Azure KeyVault användes under projektet för bildhantering och lagring genom staging, där båda delarna är viktiga för applikationens lösning vad gäller säkerhet och kostnadseffektivitet. API:er som använts under projektet är TinyPNG API och OpenAI API.

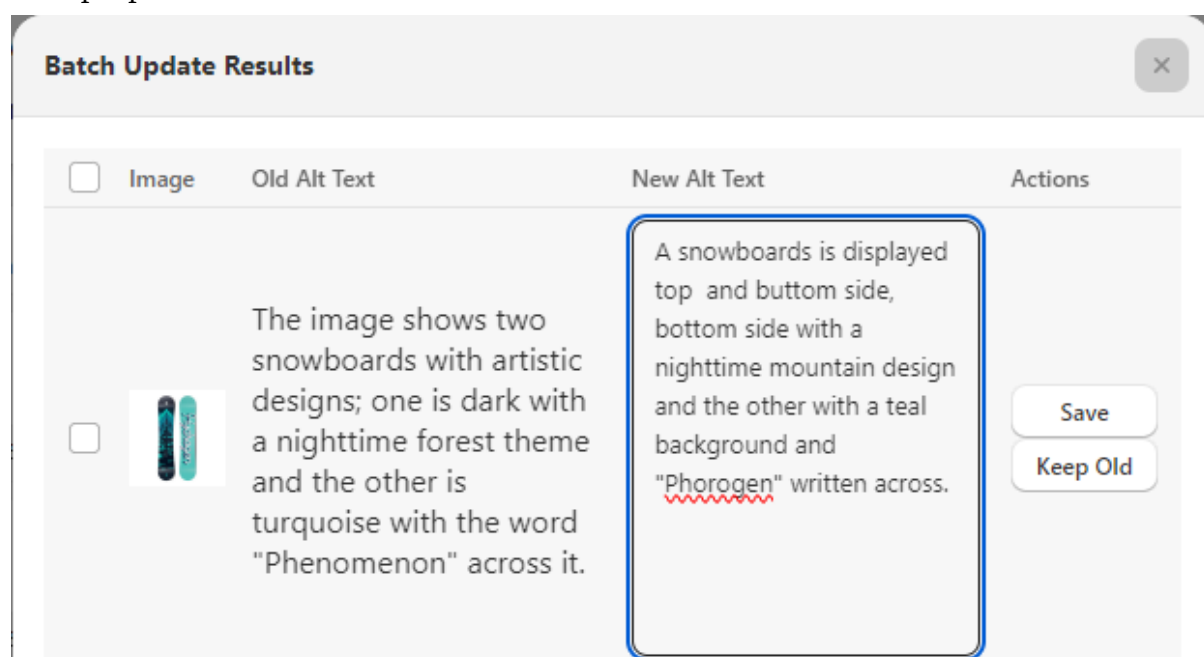
Slutligen så utvärderades projektet genom integrationstester och användartester inom företaget.

## 3. RESULTAT

### 3.1 Alt-texter

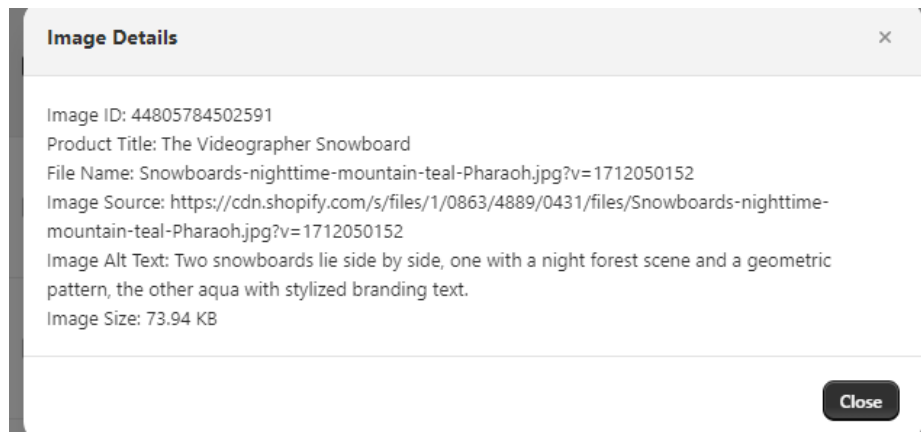
En av huvudfunktionerna i min applikation är genereringen av alt-texter för bilder. Den färdiga lösningen möjliggör automatiserat skapande av beskrivande och SEO-optimerade alt-texter, vilket förbättrar både tillgängligheten och sökmotoroptimeringen för webbplatser som använder verktyget. Vision-modellen från OpenAI användes för att ge en detaljerad beskrivning av varje bild. Denna modell analyserar bildinnehållet och skapar en noggrann textbeskrivning som fångar de väsentliga elementen i bilden.

Exempel på resultat av alt-text:



Figur 1a

Vision-modellen användes för att ge en detaljerad beskrivning av bilden. Efter att Vision-modellen hade beskrivit bilden, skickades textdatan till GPT-3.5-modellen som genererade ett filnamn med max fem ord, separerade med bindestreck (se figur 1b). Detta filnamn är optimerat för URL:er och sökmotorer (SEO).



File Name: Snowboards-nighttime-mountain-teal-Pharaoh.jpg?v=1712050152

Figur 1b

### 3.2 Azure Staging

Efter att ha verifierat att alla responser och bilder som sparades i Azure Blob Storage fungerade korrekt, insåg jag att mängden lagrad data snabbt ökade. Detta ledde till ökade kostnader. För att hantera detta implementerade jag en lösning där bilderna temporärt sparades i Blob Storage med en "Stage URL".

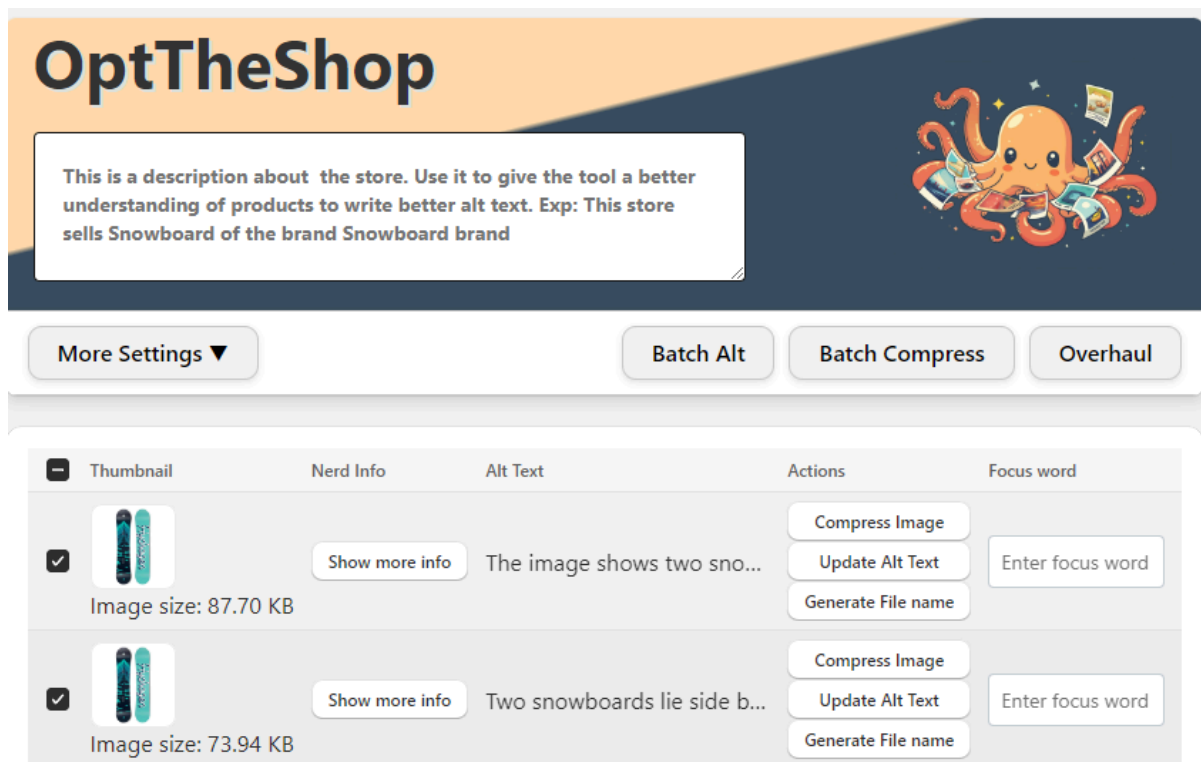
#### Process för användning av Stage URL

Bilderna laddades upp till Azure Blob Storage, där de lagrades tillfälligt. Varje bild fick en unik "Stage URL" som gjorde det möjligt att tillfälligt nå och hantera bilden. Den genererade Stage URL:en användes för att skicka bildadressen till Shopifys API. Shopify kopierade sedan bilden från Azure Blob Storage till sin egen databas för permanent lagring och hosting.

Efter att bilden hade kopierats och flyttats över till Shopify, raderades den från Azure Blob Storage. Detta förhindrade att stora mängder bilddata ackumulerades i Azure, vilket annars skulle ha lett till ökade lagringskostnader. Genom att implementera denna process med Stage URLs, blev lösningen för bildhantering både tids- och kostnadseffektiv. Denna metod möjliggjorde även säker överföring av bilder till Shopify.

### 3.3 Gränssnitt

Efter att ha undersökt liknande verktyg och säkerställt att slutresultatet skulle vara användarvänligt, uppfyller gränssnittet alla förväntningar. Gränssnittet är enkelt att navigera och innehåller endast de nödvändigaste funktionerna för att uppnå de förväntade resultaten.



Figur 2

## Design och Användarvänlighet

Genom att kombinera React och Polaris har jag kunnat skapa ett användargränssnitt som inte bara är estetiskt tilltalande och enhetligt utan också effektivt och enkelt att underhålla. Detta är avgörande för en positiv användarupplevelse, både för webbplatsens utvecklare och för användarna i sammanhanget (kunderna).

## Minimalt och Funktionellt

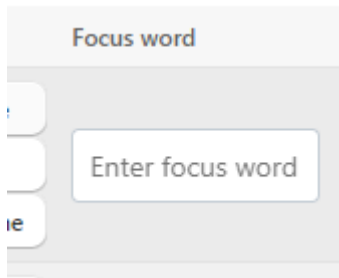
Gränssnittet är designat med enkelhet i åtanke, vilket gör det intuitivt för användare att hitta och använda de funktioner de behöver. Alla viktiga verktyg och alternativ är tillgängliga utan att överbelasta användaren med onödig information. Detta säkerställs genom att fokusera på minimalistisk design. Gränssnittet innehåller endast de mest väsentliga funktionerna som behövs för att generera alt-texter och optimera bilder, vilket hjälper användarna att snabbt och effektivt uppnå sina mål.

## Funktionalitet

Generering av alt-texten är ett enkelt formulär där användare kan ladda upp bilder och få fram det genererade textresultatet. Resultaten visas direkt på sidan, vilket underlättar processen och ger direkt feedback till användaren. Efter genereringen presenteras alt-texten där användare har möjlighet att manuellt ändra eller lägga till extra text.

Användare kan ladda upp bilder och få dem optimerade för snabbare laddningstider och bättre prestanda. Bilderna komprimeras med hjälp av TinyPNG och kan sedan laddas ner direkt från gränssnittet. Användarna kan se en förhandsvisning av bilderna med de

genererade alt-texterna innan de sparas eller används, vilket ger dem möjlighet att göra justeringar vid behov.

A screenshot of a web interface showing a section titled 'Focus word'. Below the title is a text input field with the placeholder text 'Enter focus word'. To the left of the input field, there are several small, partially visible buttons or icons.

Varje bild har ett inputfält där användare kan skriva in ett ord som bilden ska fokusera på. Detta ord skickas med i instruktionerna till OpenAI-modellen för att få en ännu mer precis beskrivning. Detta bidrar till mer exakta beskrivningar och relevanta ordval.

Användarna har även möjlighet att ge en affärsbeskrivning av vad de säljer eller om det är en specifik affär.

A screenshot of a web interface for 'OptTheShop'. The header has an orange background with the store name 'OptTheShop' in white. Below the header is a white text input field with a light gray border. The placeholder text in the field reads: 'This is a description about the store. Use it to give the tool a better understanding of products to write better alt text. Exp: This store sells Snowboard of the brand Snowboard brand'.

Denna beskrivning skickas också med i instruktionerna till OpenAI-modellen för att förbättra relevansen och kontexten i alt-texten.

I enlighet med projektets syfte och mål så har resultaten visat sig uppnå de krav som ställdes innan applikationen utvecklades.

## 4. DISKUSSION

I detta kapitel reflekterar jag över arbetet med applikationen, delar mina egna åsikter och erfarenheter samt ger förslag på hur arbetet kan utvecklas vidare. Det är viktigt att vara öppen med vad som inte fungerade som förväntat och hur detta påverkade utvecklingsprocessen. (Schwaber & Sutherland, 2024)

### 4.1 Projektets framgångar

Genom att generera beskrivande filnamn och alt-texter som exakt reflekterar bildinnehållet, förbättras bildens synlighet på sökmotorer såsom Google. Beskrivande filnamn och alt-texter hjälper sökmotorer att förstå bildens innehåll, vilket i sin tur kan leda till högre sök-rankning och högre upptäckbarhet. Fortsättningsvis kan detta leda till att webbplatser kan nå ut till en bredare publik.

Alt-texter är, som nämnt tidigare, kritiska för tillgängligheten på webbplatser. De gör det möjligt för synskadade användare att få en beskrivning av bildinnehållet genom skärmläsare. Genom att automatisera genereringen av detaljerade alt-texter säkerställer min applikation ett effektivare arbetsflöde för både webbutvecklare och content managers.

#### Fördelar med Stage URL Metoden

##### 1. Kostnadseffektivt

Genom att använda Azure Blob Storage för temporär lagring och sedan överföra bilderna till Shopify för permanent lagring, kunde jag kraftigt minska kostnaderna för långvarig datalagring i Azure.

##### 2. Effektiv hantering av bilddata

Metoden säkerställer att stora mängder bilddata inte behöver lagras i Azure under lång tid, vilket effektiviserar hanteringen av bilddata.

##### 3. Säkerhet

Genom att låta Shopify hantera den långvariga lagringen av bilder, kunde jag säkerställa att bilder som inte följer regler och riktlinjer snabbt kunde identifieras och tas bort. Detta minskade risken för att olagliga eller olämpliga bilder skulle ligga kvar i systemet.

##### 4. Automatisk Rensning

Efter att Shopify hostade bilden, raderas den automatiskt från Azure Blob Storage. Detta rensnings-steg säkerställer att endast nödvändiga och godkända bilder fanns kvar på plattformen.

## Användarupplevelse

Genom att hålla gränssnittet enkelt och fokuserat på kärnfunktionaliteten har jag skapat ett verktyg som erbjuder ett nytt alternativ till existerande verktyg. Till skillnad från tidigare alternativ erbjuder min applikation specificerade funktioner som är relevanta inom e-handel och webbutveckling. Användarnas behov och förväntningar har varit centrala i designprocessen, vilket har påverkat mina designval genom hela projektet.

Optimering av bilderna minskar bildstorleken och förbättrar prestandan på Shopify-butikerna genom snabbare laddningstider och mindre lagringsåtgång. TinyPNG API:et bidrog till att optimera användarupplevelsen genom att både minska bandbredden och lagringskostnaderna samtidigt som bildkvaliteten blev oförändrad.

Feedback från användarna användes för att finjustera design och funktionalitet, vilket resulterade i en väl uttänkt applikation med mindre utrymme för felhantering och systemfel. Företagets medarbetare deltog aktivt i de användartester som hölls, vilket gav värdefulla insikter om hur applikationen upplevdes av olika typer av användare. Denna process hjälpte till att identifiera eventuella hinder eller komplikationer som användare kunde stöta på, vilket gjorde det möjligt att åtgärda dessa problem innan den slutliga lanseringen.

## 4.2 Utmaningar och hinder

En av de största utmaningarna under projektet var att jag var tvungen att arbeta med flera olika delar samtidigt för att få saker att fungera. Det var en hel del hopp mellan olika delar av projektet, vilket gjorde det svårt att upprätthålla ett konsekvent arbetsflöde. I efterhand inser jag att det hade varit bättre att först säkerställa att API:et fungerade exakt som jag ville innan jag började hantera data i frontend.

Strukturen i frontend-koden kunde ha varit mycket bättre. Ett stort fokus låg på att göra applikationen så skalbar och lättanvänd som möjligt, men på grund av tidsbrist och behovet av att lösa akuta problem blev koden ostrukturerad på vissa ställen. Som en konsekvens av det, var jag tvungen att gå tillbaka och göra ändringar i tidigare implementeringar när vissa funktioner lades till i efterhand. I framtiden skulle en mer robust och genomtänkt struktur för frontend-koden underlätta underhåll och vidareutveckling. Att ha en mer flexibel och modulär kodbas skulle ha minskat behovet av sådana omarbetningar.

Ytterligare en stor utmaning var övergången från Gadget version 0.3 till 1.0, vilket orsakade problem med sparandet och hostingen av bilder på Shopify. Detta uppgraderingsproblem påverkade tidsplanen och tvingade mig att omprioritera vissa delar av projektet. Trots dessa problem var det lärorikt att arbeta med JavaScript och React, även om dessa är mina svagare områden. Att ta sig an denna utmaning visade sig vara värdefull för min personliga och professionella utveckling. Trots utmaningar och de hinder jag stötte på under projektet så upplever jag att mina lösningar når upp till de krav som sattes i början av projektet.



### 4.3 Förslag på vidareutveckling

För framtida projekt skulle en bättre strukturerad arbetsplan med tydliga avgränsningar mellan olika arbetsområden vara fördelaktig. Detta skulle minska behovet av att hoppa mellan olika delar av projektet och göra utvecklingsprocessen mer effektiv. (Geeks for Geeks, 2024) Genom att även lägga större vikt på en robust frontend-struktur från början kan framtida underhåll och utökningar bli enklare och mindre tidskrävande.

Jag hade planer på att integrera Google Analytics API för att ge butiksägare en SEO-score och feedback, men på grund av tidsbrist och tekniska problem hann jag inte med detta. Att inkludera denna funktion i framtiden skulle ge ytterligare värde till applikationen.

En annan potentiell vidareutveckling skulle vara att implementera funktionaliteten för att generera och ändra filnamn på befintliga bilder som inte har skapats av verktyget. Även om denna funktionalitet finns i mitt externa API, tillåter inte Shopify detta i deras API. En manuell lösning via butikens gränssnitt skulle kunna vara ett alternativ.

### 4.4 Avslutning

Sammanfattningsvis har arbetet med applikationen varit en lärorik upplevelse. Trots de utmaningar jag stött på, har jag lärt mig mycket och identifierat flera områden där förbättringar kan göras för framtida projekt. Det som jag särskilt kommer att ta med mig till framtida projekt är att strukturera upp en arbetsplan från början, och se till att applikationens mest grundläggande funktioner fungerar bra innan jag går vidare till nästa steg.

Utöver det tycker jag att jag har presterat utefter mina erfarenheter och min kunskapsnivå, där jag både har fått använda mina starkare färdigheter men också utmanat mina svaga sidor.

## 5. KÄLLFÖRTECKNING

Gadget. 2024. Serverless fullstack platform for Shopify apps. <https://gadget.dev/> (Hämtad 2024-05-17).

Geeks for Geeks. 2024. Separation of Concerns (SoC).  
<https://www.geeksforgeeks.org/separation-of-concerns-soc/> (Hämtad 2024-05-17).

Microsoft. 2024. ASP.NET: Open-source web framework for .NET.  
<https://dotnet.microsoft.com/apps/aspnet> (Hämtad 2024-05-17).

Microsoft. 2024. Azure Blob Storage: Object storage for the cloud.  
<https://azure.microsoft.com/en-us/services/storage/blobs/> (Hämtad 2024-05-17).

Microsoft. 2024. Azure Key Vault: Securely manage keys, secrets, and certificates.  
<https://azure.microsoft.com/en-us/services/key-vault/> (Hämtad 2024-05-17).

Milecia, 2019. Difference Between Development, Stage, And Production.  
<https://dev.to/flippedcoding/difference-between-development-stage-and-production-d0p>  
(Hämtad 2024-05-19)

Moz. 2024. Beginner's Guide to SEO. <https://moz.com/beginners-guide-to-seo> (Hämtad 2024-05-17).

OpenAI. 2024. OpenAI API: Access to advanced AI models. <https://openai.com/api/> (Hämtad 2024-05-17).

React. 2024. A JavaScript library for building user interfaces. <https://reactjs.org/> (Hämtad 2024-05-17).

Schwaber, K. & Sutherland, J. 2024. The Scrum Guide. <https://scrumguides.org/> (Hämtad 2024-05-17).

Shopify Polaris. 2024. Design system. <https://polaris.shopify.com/> (Hämtad 2024-05-17).

TinyPNG. 2024. Developer API for image optimization. <https://tinypng.com/developers>  
(Hämtad 2024-05-17).

W3C. 2024. Web Content Accessibility Guidelines (WCAG) Overview.  
<https://www.w3.org/WAI/standards-guidelines/wcag/> (Hämtad 2024-05-17).

# BILAGOR

## Bilaga 1

