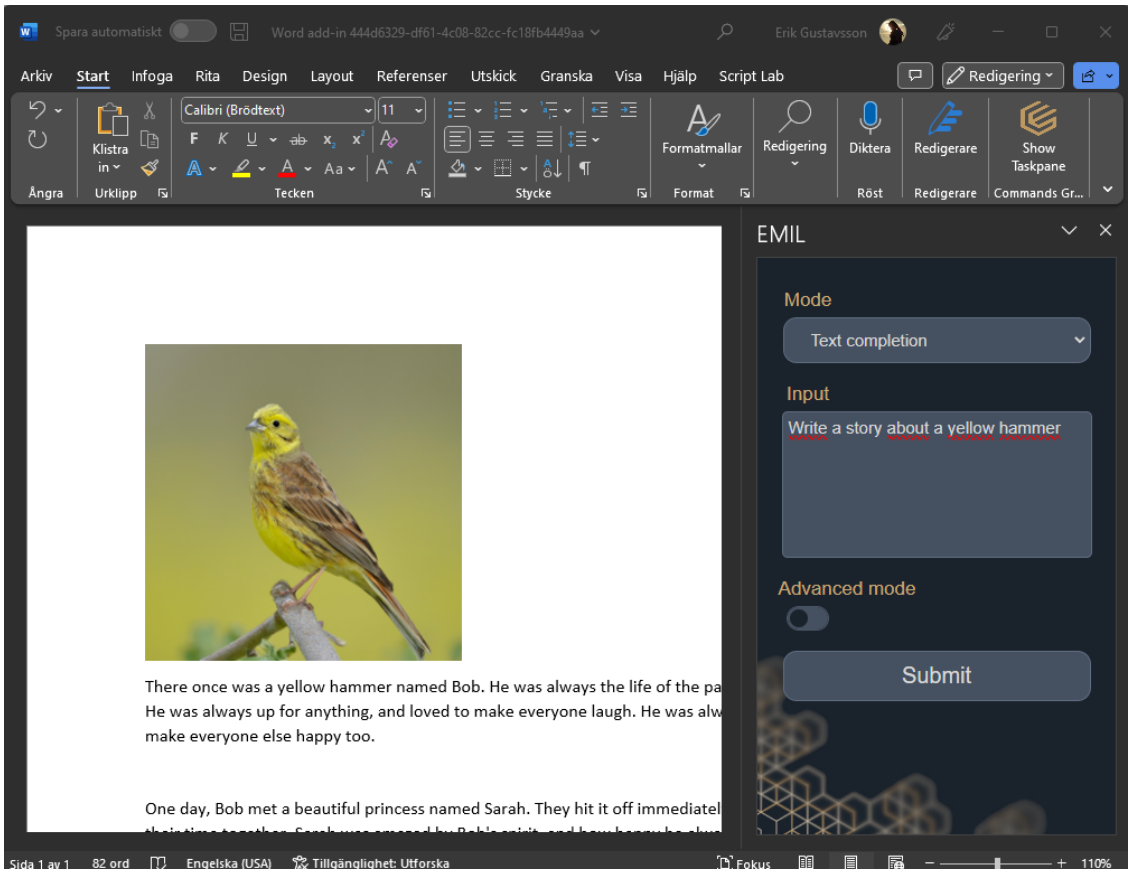


AI Word Plugin

Utveckling av ett Word plugin som använder sig av AI-text- och bildgenerering

Examensarbete inom högskoleprogrammet Datateknik

Erik Gustavsson
Oliver Brottare

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2023
www.chalmers.se

Examensarbete 2023

AI Word plugin

Utveckling av ett Word plugin som använder sig av AI-text- och bildgenerering

ERIK GUSTAVSSON
OLIVER BROTTARE

GÖTEBORGS
UNIVERSITET

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg 2023

AI Word plugin
Utveckling av ett Word plugin som använder sig av AI-text- och bildgenerering
ERIK GUSTAVSSON OLIVER BROTTARE

Handledare: Robert Krook, Data- och Informationsteknik
Examinator: Lars Svensson, Data- och Informationsteknik

Examensarbete 2023
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: En skärmdump av Microsoft Word med EMIL AI plugin som använts för att skriva en historia om en gulsparv.

Skriven i LaTeX
Göteborg 2023

AI Word plugin
Utveckling av ett Word plugin som använder sig av AI-text- och bildgenerering
ERIK GUSTAVSSON
OLIVER BROTTARE
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet

Sammanfattning

Användandet av AI bland allmänheten har ökat explosionsartat det senaste året. Många använder sig av AI-verktyg för att få hjälp med exempelvis programmering eller redigering av texter, medan många andra använder AI-verktyget av ren nyfikenhet. Syftet med detta projekt är att göra ett praktiskt hjälpmedel för skapandet av olika dokument med hjälp av AI-tekniken. Projektet realiseras med hjälp av Microsoft Word och OpenAI:s API. Programspråken som projektet är skrivet i är Javascript, HTML/CSS och Python. Stort fokus lades på att pluginet skulle vara lätthanterligt och en enkät gjordes för att mäta detta. Resultatet visar att det mycket väl går att skapa egna program för ett mer anpassat bruk av AI. Projektet ämnade även att behandla träning av egna AI-modeller för ett ännu mer specialiserat verktyg. En första model av AI-träning implementerades men finslipning av denna fick inte plats inom tidsramen.

Abstract

The use of AI by the public has increased a lot the latest year. Many use AI-tools to help with programming, editing texts and mostly maybe out of curiosity. This project describes the creation of a practical tool for the creation of different documents using AI-technology. The project is brought to life by Microsoft Word and OpenAI's API. Programming languages this project concerns are Javascript, HTML/CSS and Python. Great effort was put into ensuring that the program was easy to use. Therefore, a study was created to measure this. The Result shows that it is very possible to create an application for practical uses. The project also intended to deal with training own AI-models in order to create a more customised tool. A first model of AI training was implemented but refining it did not fit within the time frame.

Nyckelord: AI, add-in, AI-training, software development, Javascript, HTML, CSS, Python.

Förord

Denna rapport är ett examensarbete inom datateknik på institutionen för Data- och Informationsteknik på Chalmers Tekniska Högskola.

Vi vill rikta ett tack till våra handledare på Emcap, John Torgersson & Kristina Lid samt vår handledare från Data- och Informationsteknik, Robert Krook.

Erik Gustavsson, Oliver Brottare, Göteborg, Juni 2023

Begrepp och beteckningar

Nedanför är en lista på begrepp och beteckningar som har använts i denna rapport:

Add-in	Microsofts benämning på plugins till Office-program
AI	Artificiell intelligens
API	Application programming interfaces - Gränsnitt av verktyg och protokoll som program kan följa för att kommunicera med varandra
Plugin	Tilläggsprogram till ett program
Word	Microsoft ordbehandlingsprogram i office-paketet
Machine-learning	Maskininlärning
Backend	Mjukvarutveckling av det bakomliggande i program, alltså saker som logik, struktur, datahantering mm.
Frontend	Mjukvarutveckling med fokus på det visuella, det som syns av användaren
Interpreterande programspråk	Programspråk skapade för en interpreterande miljö där koden tolkas samtidigt som programmet körs till skillnad mot kompilerande programspråk
MVP	Minimal Viable Product - Det mål som minst ska uppnås inom ett scrum-projekt
Prompt	Text en AI matas med för att få ut önskat innehåll
Run-time	Tidperioden när ett program exekverar
Toggle-knapp	GUI-symbol som avbildar en vippströmbrytare för att visa ”på” och ”av” status

Innehåll

Figurer

1
Inledning

Följande kapitel går igenom projektets bakgrund, syfte, mål och avgränsningar.

1.1 Bakgrund

AI är något som diskuteras mycket inom datavetenskapen på grund av alla möjligheter som finns inom området. Genom att analysera stora mängder data med intelligenta och iterativa bearbetningsalgoritmer kan en AI-modell uppföra sig människolikt. AI-modeller har även utmaningar, t.ex. när det gäller träning på existerande data och information [1]. Vid sådan träning blir den utdata som den kan ge begränsad till sådant som redan har hänt, istället för att kunna ge nya unika tankar om hur det borde eller skulle kunna vara [2]. En AI-modell behöver stora mängder data för att fungera vilket inte alltid är lätt att få tillgång till [3]. En unik implementation av AI behöver även unik data vilket försvårar det ännu mer. En AI är svår att konstruera på ett säkert sätt. Manipulering av träningsdata eller inkorrekt inmatning till en AI kan leda till sabotage av algoritmerna och ge en helt fel tolkning. Ett exempel på detta är chattbotten Tay som utvecklades av Microsoft och blev tillgänglig på Twitter [4]. Denna chattbot hade flera filter för att den inte skulle skriva olämpliga saker, men under de första 24 timmarna började den skriva olämpliga saker efter att vissa personer hade utnyttjat en svaghet.

Många AI-drivna applikationer finns tillgängliga för allmänheten. En del av dessa applikationer rör AI-genererade texter. Bland de mest välrenommerade återfinns ChatGPT, Simplified, AI-writer för att nämna några [5][6][7]. Dessa textgeneratorer kan bland annat skapa nya texter utifrån nyckelord och även förändra befintliga texter. De kan exempelvis rätta grammatiska fel, stavfel och ändra textstil. Dessa generatorer fungerar oftast genom att användaren kopierar in text eller skriver ett antal nyckelord i ett fält. Sedan genereras en ny text som resultat på inmatningen.

Det är mycket resurskrävande att skapa en egen AI från grunden med motsvarande kapacitet som de skapade av stora företag. Exempelvis så investerade Microsoft 13 miljarder dollar i OpenAI som ligger bakom ChatGPT [8]. För att utnyttja kraften av AI i ett eget program kan man istället använda vissa befintliga AI:n med hjälp av Application programming interfaces (API). API gör det möjligt att överföra data mellan egen applikation och annat program, exempelvis en kraftfull AI.

1.2 Syfte

Syftet med detta arbeta är att skapa en AI-driven textgenerator i en ordbehandlare via ett plugin. Genom att tillgängliggöra AI inne i en ordbehandlare kan användare få stor hjälp genom textgenerering. AI-textgenerator ska vara ett hjälpmedel för att kunna få fram en textkropp som sedan kan redigeras för att slipa fram en önskad text. Verktyget hjälper användaren att spara tid, komma igång med en text samt få en inblick i hur en typisk text inom området kan vara strukturerad.

1.3 Mål

Målet med projektet var att ta fram en prototyp av ett plugin-program inuti Microsoft Word. Detta plugin skulle låta användaren skriva in nyckelord som AI:n sedan skrev ut som text. Målet var uppfyllt när en användare kunde skriva in 3-5 nyckelord i plugin-verktyget och få ut en textmall som var relaterat till dessa. Ifall användaren efterfrågar det, ska texten exempelvis ha rubrik och underrubriker med tillhörande textkroppar som alla är relaterade till vad användaren skrev in. I Minimal Viable Product (MVP) gällde det att texten var av typen artikel. Målet var även att lägga till fler funktioner om tid fanns exempelvis bildgenerering och om det var möjligt med det AI-verktyg som användes. Detta plugin skulle ha ett användarvänligt GUI med inställningar för att anpassa det efter hur användaren vill ha det. Användarvänlighet syftade på att det skulle vara lättnavigerat för en ny användare och produktmässigt accepterat av produktägaren (PO). Minst två personer utan kännedom om programmet skulle testa användarvänligheten för att få en indikation huruvida programmet var lättnavigerat eller inte. Detta genomfördes med hjälp av enkäter för att mäta olika områden av arbetet. När dessa personer och PO ansåg att programmet var lättnavigerat var GUI-målet klart.

När ovanstående var klart kunde träning av verktyget på eget material undersökas för att visa att konceptet kunde anpassas till olika ändamål. Verktyget skulle då skriva ut anpassad text vilket skulle vara användbart för organisationer eller företag som behandlar vissa texttyper och använder särskilda fackspråk. Ett annat exempel på vad den skulle kunna göra är att läsa av text från ett företags hemsida för att sedan kunna svara på frågor och skriva om företaget. Den skulle då urskilja sig från de vanliga AI-modellerna som ibland saknar kunskap för att kunna skriva en relevant text. Detta räknades dock som en extrauppgift och var något som genomfördes när det fanns tid över.

Frågeställningarna löd:

- Kommer plugin-programmet vara praktiskt att använda för en användare med syftet att skriva ut en textmall?
- Kommer användare tycka att plugin programmet är lätthanterligt?
- Kan OpenAI API tränas på egen text så att den blir användbar för anpassade texter?

1.4 Avgränsningar

Projektet kommer att begränsas till att enbart fungera i Microsoft Word. Projektet planeras även att begränsas till att fungera på installationen av Word på datorer med Windows operativsystemet och fungera i online-versionen eller desktop-versionen av Word. Det kommer endast förekomma kontroller av att verktyget fungerar på text skriven på engelska och ifall text på andra språk matas in kan resultatet variera från vad användaren förväntar sig.

2
Metod

Följande kapitel går igenom förberedelserna som gjordes innan mjukvaruutvecklingen och vilken roll Emcap hade under projektet. Sedan beskrivs vilka verktyg som användes och hur de användes.

2.1 Arbetsmetod

Projektet utfördes med projektledningsmetoden Scrum. Scrum är ett flexibelt arbetssätt där arbetet är uppdelat i intervaller som kallas ”sprints”. Uppdelningen gör det mer flexibelt vilket passar bra för mjukvaruutveckling då oförutsägbara vändningar i planeringen ofta förekommer. Scrum valdes därför som arbetssätt och vår produktägare (PO) blev då våra handledare på Emcap, John Torgersson & Kristina Lid. De hade som jobb att se värdet i de olika delarna av produkten och komma upp med prioriteringar och ändringar under projektets gång. Detta skulle genomföras genom möten som hölls veckovis där arbetet presenterades.

Projektet planerades genom att det först undersöktes hur arbetet skulle utföras. Efter detta skulle sprints skapas med olika mål som skulle uppnås till varje sprint. När detta väl var genomfört skulle arbetet till sist påbörjas. Ifall det visade sig att det fanns tid över så var det planerat att utforska möjligheten att träna en AI som man sedan kan använda i plugin-programmet.

2.2 Verktyg

Följande delkapitel går igenom verktygen som användes under projektets gång. Kapitlet beskriver vad varje verktyg kan göra och varför det valdes för detta arbete.

2.2.1 Yeoman generator

”Yeoman Generator for Office Add-ins” (Yo Office) användes för att skapa en grundläggande plattform för utvecklingen av plugin-verktyget [9]. Denna grundläggande plattform skapade ett enkelt demo-plugin inuti Word. Den satte även upp kod och de grundläggande konfigureringar i Node packet Manager (Npm) som behövdes för att plugin-programmet skulle kunna startas inofficiellt utan att behöva ligga uppe på Words ”Add-in” katalog online. Detta gjorde att arbetet snabbt kunde komma igång.

2.2.2 Integrated development environments (IDE)

Som utvecklingsmiljö valdes Visual Studio/Visual Studio Code. Dessa hade bra stöd för Javascript, CSS, HTML, Python samt är utvecklat av Microsoft och rekommenderades av Microsoft för utveckling av Office plugins. Visual Studio har många av de vanliga funktionerna hos en IDE som används under utveckling som debugger, stöd för flera språk, inbyggt stöd för GitHub och möjligheten att installera tillägg som underlättar utvecklingen. Under detta arbete användes den som utvecklingsmiljö för att skriva all kod i programmet.

2.2.3 Microsoft Word

Plugin-programmet som utvecklades skulle kunna köras i Microsoft Word. Under utvecklingen användes därför Microsoft Word för att kontrollera ifall funktioner fungerade. Det var även möjligt att genomföra debugging vilket underlättade när fel uppstod. Detta utfördes både i webbläsarversionen av Word och Word desktop-programmet.

2.2.4 Figma

Figma valdes som designverktyg för att göra skisser av plugin-programmet [10]. Det valdes eftersom det fanns tidigare erfarenhet av verktyget. Eftersom det går snabbt att skapa olika skisser i verktyget är det hjälpfullt att använda Figma eftersom en överblick ges av hur designen kan se ut. Då behöver realisering av skissen i HTML/CSS oftast bara utföras en gång eftersom man på förhand vet hur designen kommer att se ut. Figma gör det även möjligt för flera personer samtidigt att ändra och kolla på en skiss vilket gör det enkelt att diskutera fördelar och nackdelar av designen.

2.2.5 GitHub

Projektet laddas upp på en förvaringsplats online via tjänsten GitHub [11]. Genom GitHub kunde versionshistorik sparas samt arbetet delas upp och sammanfogas. Github hjälper även för att undvika dubbelarbete eller misstag, som att till exempel skriva över varandras kod. Github valdes eftersom erfarenhet med verktyget fanns sedan tidigare och att visual studio hade ett inbyggt stöd för det.

2.3 Enkät

För att utvärdera hur lätthanterligt plugin-programmet blev så skapades en enkät. Enkäten skapades i Google Forms eftersom det var ett enkelt och gratis sätt att skapa enkäter på och eftersom det inte behövdes skapas några nya konton för att använda det. Enkäten utformades på ett sätt där användaren först skulle kolla mer på de grafiska aspekterna av programmet och de hjälpmedel som lagts till i form av beskrivande texter. Efter att användaren svarat på de frågorna skulle de själva testa sig fram. Detta gjordes för att se vad användarna kunde göra utan att de fick några

instruktioner. Till sist skulle de sedan svara på några frågor som undersökte mer vad deras erfarenhet av plugin-programmet var efter att de nu fått använda det.

2.4 Konstruktionsflöden

Arbetet delades upp efter hand när det var lämpligt. Exempelvis arbetade en med frontend i HTML medan den andra gjorde funktioner i backend i Javascript, en gjorde en skiss i Figma medan den andra kopplade ihop frontend med backend och en jobbade med att utveckla AI-träning medan den andra integrerade AI-träning och plugin. Uppdelningen skedde beroende på vad varje individ var mån att lära sig och baserat på kompetenser inom olika områden.

3

Teknisk Bakgrund

I detta kapitel täcks den tekniska bakgrunden av de programmeringsspråk som användes under projektet. Kapitlet täcker även bibliotek och paket som hade stor påverkan på hur programmet fungerade.

3.1 HTML

HyperText Markup Language (HTML) är ett märkspråk som beskriver strukturen på en webbsida [12]. Det betyder att det är ett programspråk med textkoder samt vanlig text, där textkoderna inte syns när användaren läser dokumentet efter att det blivit tolkat av en webbläsare. HTML gör det möjligt att lägga in olika element som textrutor, paragrafer och knappar. I detta projekt används HTML tillsammans med CSS för att designa utseendet och JavaScript för att ge funktionalitet till de olika elementen.

3.2 CSS

Cascading Style Sheets (CSS) är ett språk som modifierar hur olika element presenteras [12] i ett strukturerat dokument. Den kan exempelvis användas för att ändra färg, storlek och textfont. I detta projekt används CSS för att ändra utseendet på elementen i plugin-programmet. Genom att använda CSS kan man skapa klasser som används av flera element samtidigt, vilket underlättar mjukvaruutvecklingen eftersom det minskar upprepning av kod.

3.3 Javascript

Javascript är ett interpreterande programspråk som kompileras under run-time [13]. Vanligaste användningsområde är främst på hemsidor och webbapplikationer för att möjliggöra interaktivitet. Javascript-koden bäddas in i HTML-koden som sedan körs av en webbläsare. Javascript-kompilatorer är inbyggda i alla större webbläsare eftersom Javascript används av en majoritet av alla hemsidor. Javascript används även av servrar och i inbyggda applikationer. I detta projekt används Javascript på det sistnämnda sättet, för att bygga in en applikation inuti MS Word.

3.4 Node.js

Node.js är en plattformsoberoende- JavaScript-runtime miljö [14]. Node.js används för att köra Javascript utan att en vanlig webbläsare behöver användas. Den kompilerar i realtid och är baserat på öppen källkod. Den har även stöd för asynkron kommunikation vilket är positivt när man jobbar med funktioner som kan ha olika fördröjningar för att undvika att processer blockeras. I detta projekt används den för att sätta upp en lokal server för att köra plugin-programmet.

3.5 Webpack

Webpack är en så kallad ”static module bundler” och används för att packa ihop filer i ett program. Webpack bygger upp en ”dependency graph”, vilket är ett nätverk av alla filer som är beroende av varandra inom applikationen. Detta är användbart eftersom många moduler och färdiga små bibliotek ofta används för program skrivna i Javascript och dessa behöver hänvisas till varandra. Webpack packar ihop dessa moduler till en eller några få paket som sedan laddas in av webbläsaren eller Node.js servern. Detta gör så att varje liten modul finns med under run-time [15].

3.6 NPM

NPM är en pakethanterare för Node.js som tillåter enkel delning av paketerade moduler av kod [16]. NPM-registret innehåller även paket med öppen källkod för Node.js, mobilapplikationer och många andra saker som JavaScript kan behöva stöd för. I detta projekt används det för att installera Yeoman generator office-paketet, installera olika bibliotek till node.js samt för att köra ett startscript med kommandot ”npm start”, vilket startade node.js servern.

3.7 Python

Python är ett interpreterande, objektorienterat- högnivåspråk [17]. Python använder en syntax som är enkel att förstå och har ett brett stöd av olika programvarubibliotek. Python används även ofta som ett ”expanderingsspråk”, för att addera ytterligare funktioner eftersom Python har ett bra stöd för att integrera med övriga delar från en applikation. I detta projekt används Python för att använda OpenAIs Python-bibliotek samt för att skapa en separat lokal server för AI-träning.

3.8 HTTP

Hypertext Transfer Protocol (HTTP) är ett protokoll som hanterar kommunikation mellan klient och server [18]. HTTP ingår som protokoll på internets applikationslager och ansvarar för förfrågningar och överflyttning av data. HTTP är fundamentalt

för kommunikation över internet men kan även användas för att kommunicera mellan lokal server och applikation. Genom HTTP-protokollet kan ett objekt skickas mellan klient och server. Ett objekt är en fil och kan innehålla vad som helst.

3.9 OpenAI API

”OpenAI” är företaget OpenAI:s API för att möjliggöra access till OpenAI:s AI från eget program. Den valdes eftersom Emcap erbjöd finansiellt stöd för den och eftersom det fanns väldokumenterade guider till den. Biblioteket möjliggör access till färdiga funktioner som kan anropas för att skicka en begäran till en av OpenAI:s databaser som returnerar ett svar. När man genomför dessa anrop kan man ha olika värden på flera parametrar för att få olika svar. Detta användes bland annat för att byta mellan olika AI-modeller. De AI-modeller som går att använda i detta plugin är davinci-003, curie-001, babbage-001 och ada-001. Dessa modeller skiljer sig åt i hur avancerade uppgifter de kan utföra, hur snabbt de kan svara och hur mycket det kostar för att generera ett svar. För att kunna använda databasen behöver användaren en API-nyckel som är kopplad till ett betalkonto hos OpenAI [19]. OpenAI API används i detta plugin för att generera text, bilder och träna AI-modeller.

3.10 Python bibliotek

Här följer en kort beskrivning av de olika bibliotek som användes i Python-programmet i projektet.

3.10.1 Flask

Flask är ett framework skapat för att utveckla webbapplikationer med Python. Framework innebär att det är en slags grundstruktur som man kan använda för att bygga vidare någonting på. Flask är ett Web Server Gateway Interface (WSGI) och specificerar hur serverar kommunicerar med webbapplikationer och hur de båda kan hantera protokollförfrågningar [20]. Med hjälp av Flask kan ett Python-program agera server och ta emot HTTP-förfrågningar samt skicka tillbaka HTTP-respons.

3.10.2 Flask-CORS

Flask-CORS är ett tillägg till Flask för att hantera Cross Origin Resource Sharing (CORS). CORS behöver hanteras eftersom ”same-origin policy” gäller som standard. Detta innebär att ett script som körs på en server har begränsade möjligheter att kommunicera med resurser som har sitt ursprung från en annan server. Same-origin policy beskrivs av The Internet Enginering Task Force (IETF) som en policy för webbdatasäkerhet [21]. Den existerar för att göra det svårare för fientligt inställda att utnyttja bland annat cookies för att få tillgång till känslig information [22].

3.10.3 JSONlines

JSONlines är ett bibliotek som används för att formatera om ett JSON-objekt till JSONline-objekt. JSONline är ett format där ett JSON-objekt finns per textrad. JSONline är ett smidigt sätt att skicka data mellan webbapplikationer [23].

3.10.4 Pandas

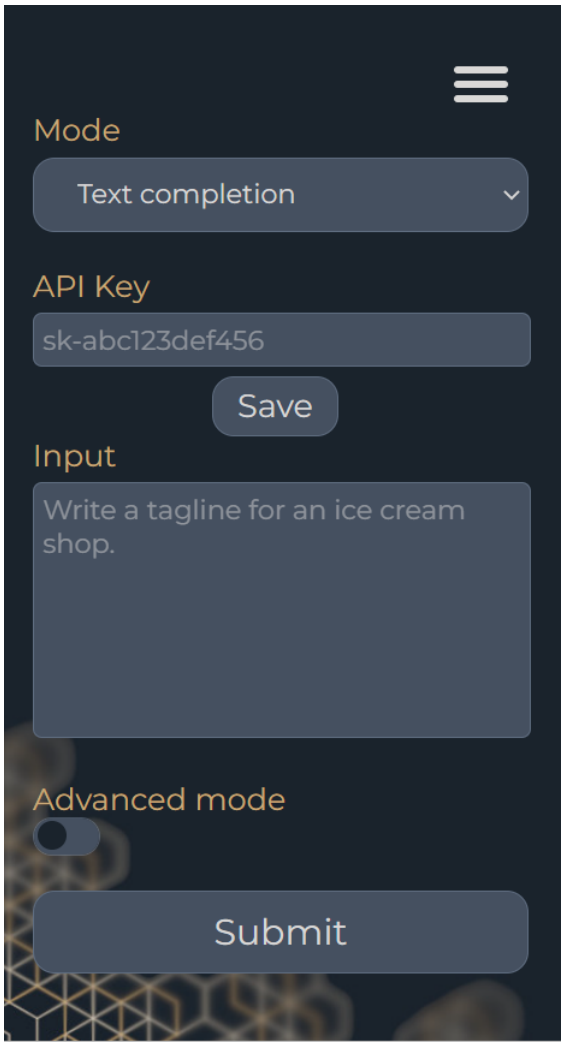
Pandas är ett open source-bibliotek till Python som förenklar hanteringen av data i tabellform. Det kan vara användbart om man jobbar med exempelvis databaser eller kalkylblad [24].

4

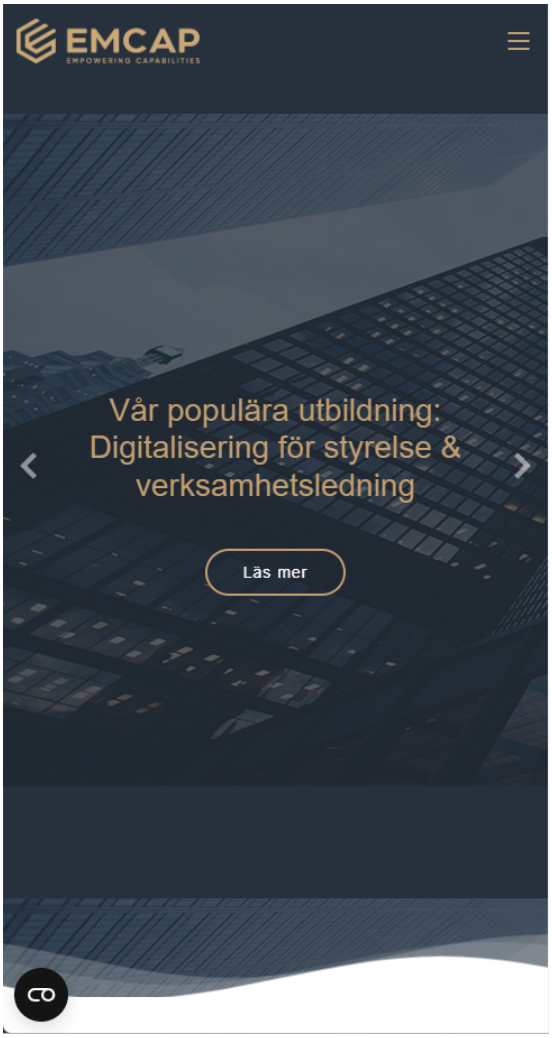
Genomförande

Följande kapitel beskriver utförandet av arbetet och lyfter fram viktiga moment som uppstod. Kapitlet börjar med att beskriva den grafiska designen. Sedan beskrivs backend-designen på projektet och hur det byggdes upp. Efter det beskrivs det hur programmet tillgängliggjordes för användaren och slutligen AI-träningsmomentet.

4.1 Grafisk design

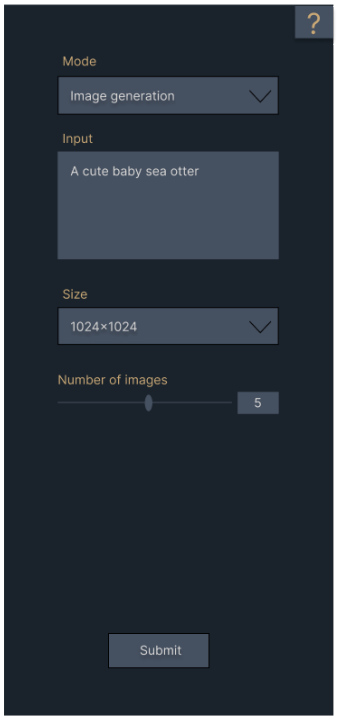


Figur 4.1: Grafisk Design

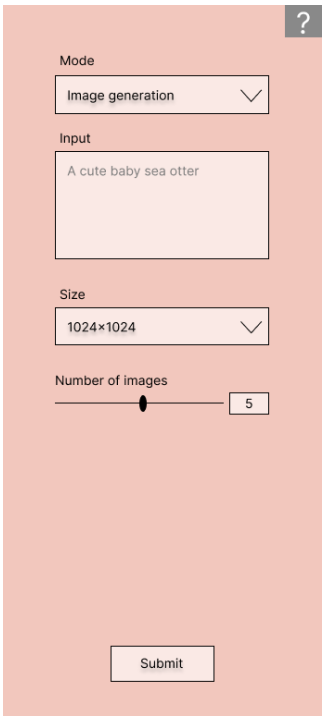


Figur 4.2: Emcaps hemsida

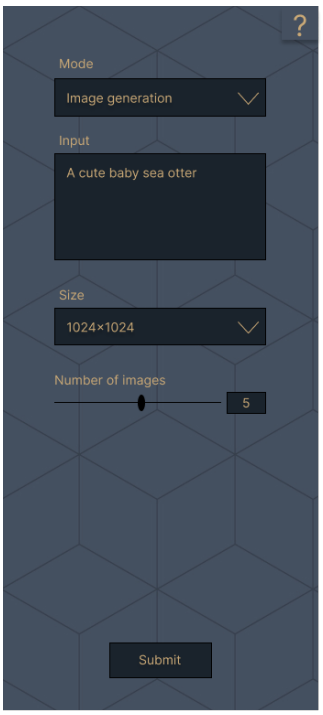
Färgvalet gjordes utifrån Emcaps grafiska profil. En mörkare design valdes eftersom Emcaps färgval i huvudsak är mörkare varpå det blev enklare att skapa en design som matchar Emcaps varumärke. I Figur 4.1 syns designen som blev vald. Figur 4.2 visar Emcaps hemsida som var inspiration till den design som valdes. Guldfärg valdes till rubriker, en mörkare blåaktig färg som bakgrund samt en ljusare blå färg till rutor för att ge en framhävande effekt. För att koppla ihop det ännu tydligare med Emcaps varumärke används i bakgrunden ett hexagonmönster som återfinns i Emcaps design. Hexagonmönstret gjordes oskarp så att inte skarpa linjer skulle konkurrera med text och göra applikationen svårläst. Typsnittet ändrades till Monteserrant font efter Emcaps grafiska profil. I alla rutor rundades kanterna av. Detta för att skapa ett mer mjukt intryck samt för att passa in i Words tema som också har rundade rutor. Små animationer i form av färger när man håller över knappar och att man visuellt kan se när en knappar trycks ner användes också för att ge en mer responsiv känsla från plugin-programmet. För att skapa designen gjordes först prototyper i Figma, sedan valdes designen ut och implementerades i HTML/CSS.



Figur 4.3: Mörk design



Figur 4.4: Rosa design

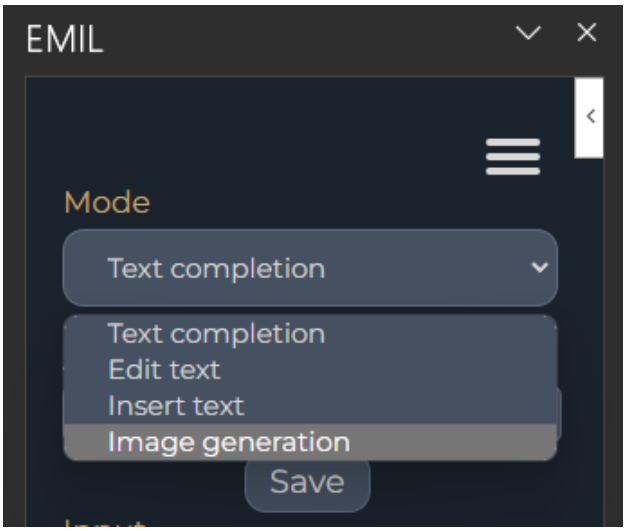


Figur 4.5: Ljus design

Några alternativa designer skapades också. I figur 4.3, 4.4 och 4.5 syns de tre ursprungliga designerna. Av dessa ansågs den mörka designen vara den bästa eftersom den skapade en bra kontrast i Word och fick plugin-programmet att både stå ut som en egen del, men ändå passa in med resten av färgschemat. Den rosa designen hade också stuckit ut men var svårare att koppla till varumärket då den oftast används insprängd som en kontrast till det övriga guld/mörkblå temat. Emcaps hemsida var den största inspirationen till designen på plugin-programmet. Den ljusa designen blev lite svårläst på grund av guldbokstäverna.

4.2 Plugin-programmets funktioner

Följande delkapitel beskriver programmets funktioner som är tillgängliga för användaren. Se figur 4.6 q



Figur 4.6: Rullgardinsmenyn ”Mode” med möjliga val

4.2.1 Text completion

”Text completion” är en huvudfunktion i OpenAI:s API och tillhandahåller en lättanvänd funktion som ger många möjligheter. Med completion kan användaren specificera vad användaren vill ha för respons från AI-modellen. En användare kan exempelvis skriva: ”Write a technical report about...” eller ”Write a public announcement about... with five paragraphs and a title”. Eftersom användaren kan skriva in precis vad den vill ger denna funktion möjligheter för användaren att låta AI:n skapa precis vad användaren efterfrågar. Denna funktion kan användas för att skapa textmallar vilket var ett av huvudmålen för projektet. För att använda Text completion skriver användaren in en prompt i textfältet och klickar på submit, sedan klistras utskriften in i Word-dokumentet.

4.2.2 Edit text

”Edit Text” är en funktion som möjliggör AI-redigering av text. Möjliga användningsområden är exempelvis för att göra en text mer formell, lättläst, få ett annat språk, rätta grammatik med mera. För att använda edit text behöver användaren kopiera in sin text i ett textfält i plugin-programmet, samt skriva instruktioner i ett ytterligare textfält som är tillgänglig när edit text är vald som funktion.

4.2.3 Insert text

”Insert text” är en funktion som möjliggör AI-generering av text med hänsyn till befintlig text. I prompten till denna funktion skriver användaren in två textstycken,

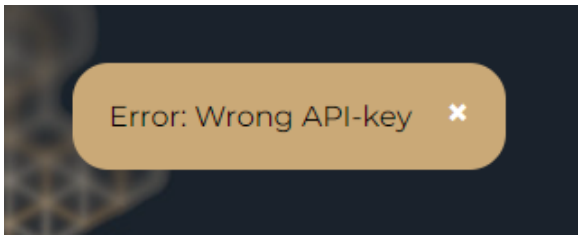
separerade med ett nyckelord ”[insert]”. AI:n skriver sedan in text som binder ihop de två styckena. Det AI:n skriver är helt beroende på vad det står i de enskilda styckena. Vill användaren exempelvis få ett annat avslut på en berättelse kan användaren skriva ett önskat avslut efter ”[insert]” och låta AI:n skriva hur berättelsen kommer fram till detta avslut.

4.2.4 Image generation

”Image generation” skapar bilder utifrån användarens prompt. Dessa bilder kan användas i dokumentet för att visualisera något innehåll användaren vill lyfta fram. För att skapa en bild skriver användaren alla detaljer som önskas i bilden och sedan klickar användaren på submit-knappen. Bilden skrivs sedan ut och hamnar längst ner i dokumentet. Användaren kan välja på bildstorlekarna 256x256 pixlar, 512x512 pixlar och 1024x1024 pixlar.

4.2.5 Ytterligare funktioner

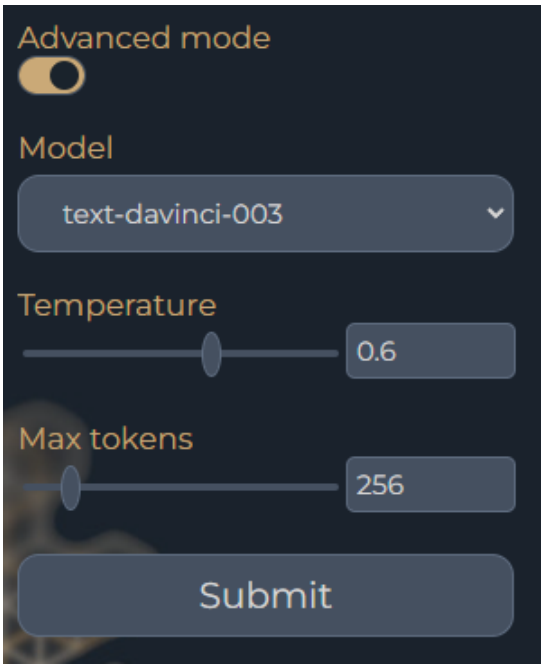
Under utvecklingens gång efterfrågades några ytterligare funktioner som implementerades.



Figur 4.7: Error meddelande

4.2.5.1 Pop-up felmeddelande

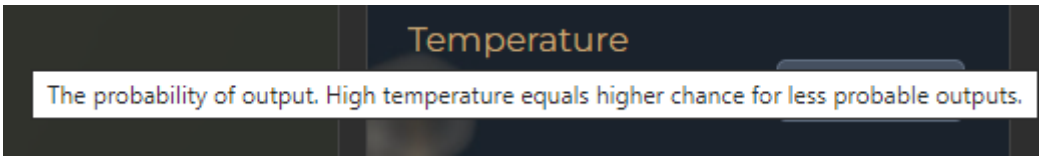
Under ett möte poängterade en av handledarna på Emcap att det hade varit användarvänligt om användaren fick respons på om något fel inträffade. Ett pop-up meddelande utvecklades därför. Popup-meddelandet är programmerat i CSS och har ett kryss så att man kan ta bort den. Felmeddelandet genereras när funktionen writeError() används i Javascript. Några felkoder har översatts och skrivits ut till text. Exempelvis error: 500, som betyder att OpenAI servern fick något problem när den försökte hantera AI-förfrågan. (Detta skedde några gånger under utvecklingen när OpenAI-serven hade driftstopp). Även fel API-nyckel översattes eftersom det var ett vanligt förkommande fel. Övriga fel skrivs ut med det felmeddelanden som genereras som standard. I Figur 4.7 syns resultatet.



Figur 4.8: Advanced mode med toggle-knapp

4.2.5.2 Advanced mode

Plugin-programmet är utvecklat så att användaren själv kan ställa in max_tokens, temperature, vilken slags AI-modell samt storlek på bild. Temperatur är en parameter som kontrollerar ifall AI:n ska returnera ett sannolikt eller mindre sannolik respons. En låg temperatur är bra när det endast finns ett svar användaren önskar medan en hög temperatur är bättre om användaren vill skapa något mer kreativt, exempelvis en berättelse. Max_tokens sätter en gräns på hur långt svaret kan bli, där alla modeller kan ha en gräns på upp till cirka 8000 bokstäver. Då grundtanken var att ha ett lättanvänt program skapades ett ”Advanced mode” som kan aktiveras på knappen under textfältet för prompts. Den består av en toggle-knapp, programmerad i CSS, som tar fram en mer avancerad meny. Vid knapptryck på toggle-knappen göms eller visas menyn med hjälp av Javascript-kod. Menyn består av sliders, programmerade i CSS, samt flervalsrutor. Menyn ändras även beroende på vilket läge som är inställt. Väljs exempelvis ”Image generation”, så visas en slide där användaren kan bestämma storlek på bild. Dock ingen modell, temperature eller max_tokens då dessa har bestämda värden enligt krav från OpenAI. Advanced mode lades till för att användaren skulle kunna ha mer kontroll över det som skrivs, men toggle-knappen är avstängd från början för att nya användare inte ska bli distraherade av alla valmöjligheter. De som har mer erfarenhet kan klicka på toggle-knappen för att ytterligare funktionerna ska dyka upp. I figur 4.8 syns advanced mode för ”Text completion”.



Figur 4.9: Förklaringstext vid hover över ”Temperature”

4.2.5.3 Hjälptext

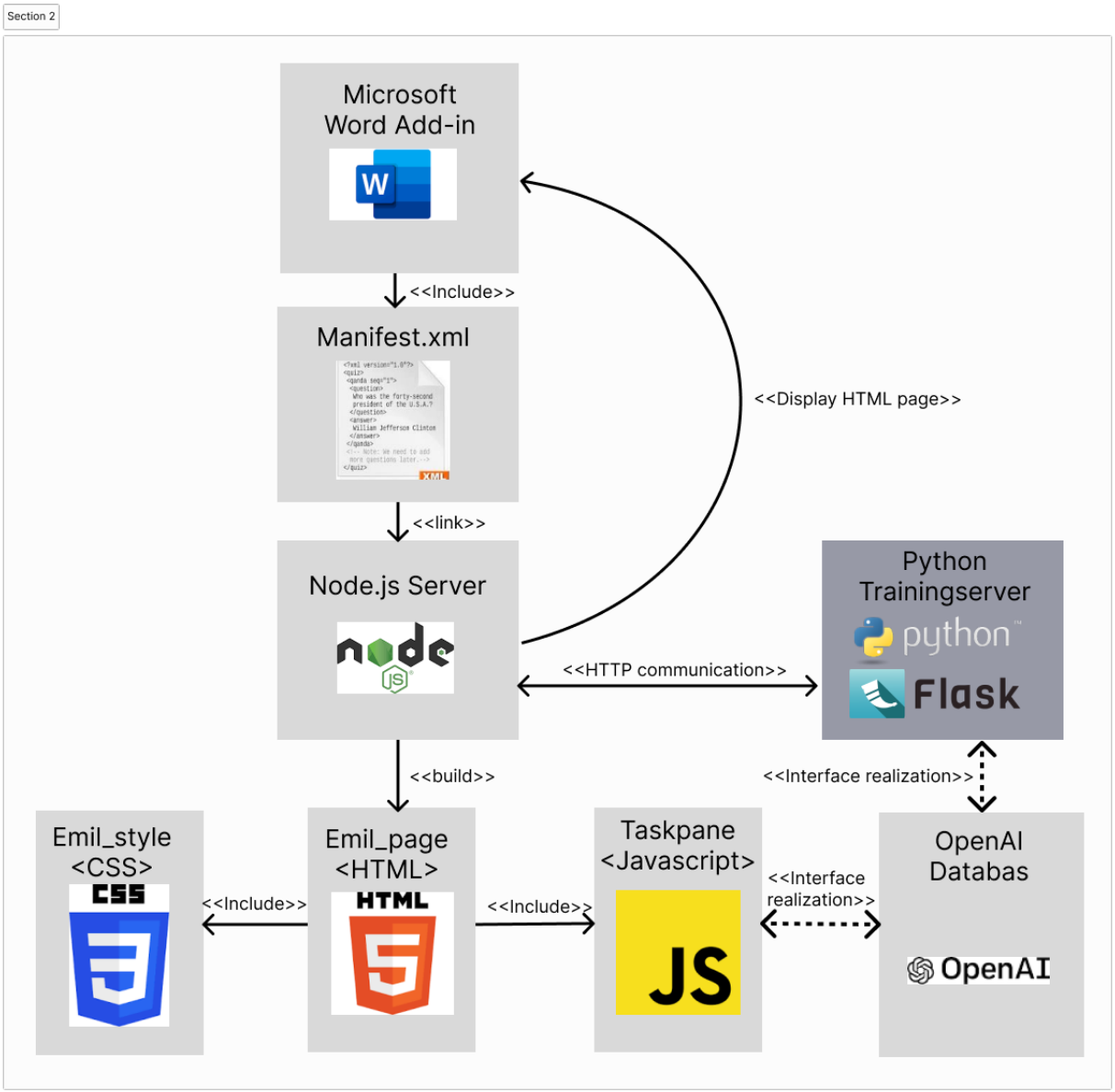
Planen var från början att ha en egen hjälpruta som förklarar hur man använder programmet. Efter lite prioriteringar valdes detta bort och ersattes av hjälptext som kommer upp när användaren ”hovrar” över rubriker i programmet. Figur 4.9 visar texten när pekaren ligger över ”Temperature”. HTML har denna funktion inbyggd när man skapar texter på en sida med textkoden <title>.

4.3 Backend Design

Följande delkapitel går igenom hur de olika delarna av programmet fungerar tillsammans och visar hur textutskrivningen är implementerad.

4.3.1 Utvecklingsmiljön

Utvecklingsmiljön skapades med Yeoman.io verktyget. Grundstrukturen består av en manifest.xml fil som Word läser in för att öppna programmet som ett plugin. Även en Javascript-fil och HTML-fil finns som möjliggör ändringar av GUI och programmeringsbara interaktioner. Ett Word-API är installerad i Javascript-filen vilket möjliggör skrivning, läsning och på andra sätt interaktion med Word-dokumentet. En Node.js server finns också som är ihopkopplad med en HTML-fil och manifest-fil.

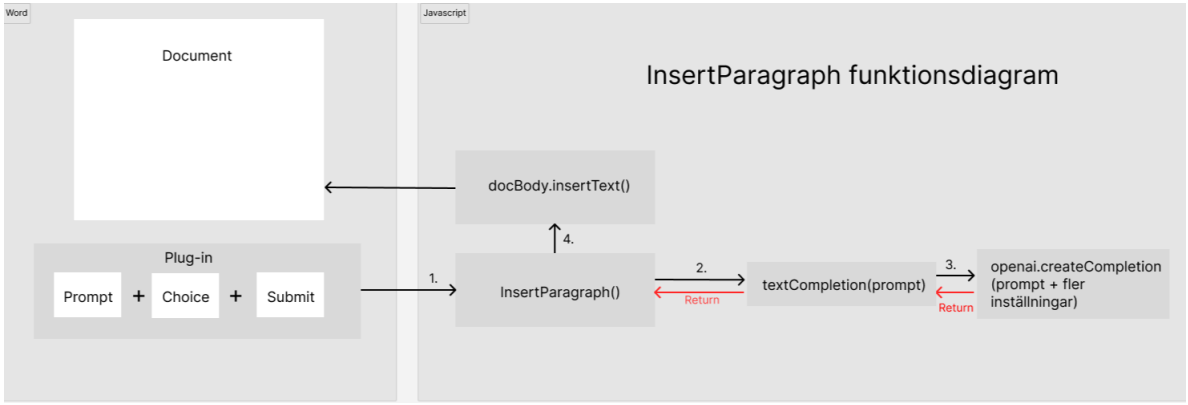


Figur 4.10: Programstruktur

4.3.2 Programstruktur

I figur 4.10 ser man hur programstrukturen är uppbyggd. Vid uppstart startas en Node.js server på localhost:3000. På denna server ligger HTML sidan Emil_page. Emil_page har Emil_style inkluderad för appens utseende och Javascript inkluderad för att möjliggöra interaktion under run-time. Javascriptet har kod som anropar OpenAI:s Databas med prompts för att få ett svar utifrån det man önskar. Vid uppstart av Word läses manifest-filen in som talar om för Word att en add-in i form av en HTML-sida ligger på localhost:3000. När Word har denna koppling visas plugin-programmet i Word. Vid knapptryck och inmatning av text körs kod genom Javascript som kan skriva ut text, bilder osv. i Word-dokumentet. Den mörkgrå rutan i Figur 4.10 är en Python-server som finns med för att möjliggöra AI-träning.

Mer om den beskrivs i stycke 4.4.



Figur 4.11: insertparagraph() - förenklad modell

4.3.3 Nyckelfunktion: Insertparagraph()

För att kunna använda OpenAI behövde plugin-programmet kunna skicka anrop till OpenAI genom dess API. I Figur 4.11 visas ett exempel på hur detta är implementerat genom att visa hur en vanlig textgenerering skapas. Modellen är avskalad för att göra exemplet enklare, Insertparagraph() har i verkligheten fler funktioner. Se Appendix A.1 för en mer komplett modell. Samma struktur som visas i detta exempel används för image generation, insert text och edit text.

1. I plugin-programmet skrivs en prompt in. Även val av AI-modell, temperatur och max_tokens kan göras av användaren. Inmatningen av all denna information hanteras av Taskpane (Javascript-filen, se Figur 4.10), i andra funktioner än Insertparagraph() som är exemplifierad här. När knappen submit klickas, startas funktionen InsertParagraph().
2. InsertParagraph() bestämmer vilket slags AI-verktyg som ska användas baserat på ”Mode” som är valt av användaren. InsertParagraph() skickar sedan prompten till funktionen textCompletion() eftersom ”Text completion” var valt i ”Mode”.
3. textCompletion() kör i sin tur funktionen openai.createCompletion(), med inskriven prompt och övriga val ifyllda. Detta är en av funktionerna från Open AI:s bibliotek. När AI-innehållet har genererats skickas det tillbaka från OpenAIs databas till textCompletion(), vilken i sin tur returnerar tillbaka till InsertParagraph().
4. I insertParagraph() skrivs texten ut i dokumentet genom docBody.insertText() funktionen, vilken är en färdig funktion från Words Javascript-biliotek. På liknande sätt fungerar ”image generation”, ”insert text” och ”edit text”.

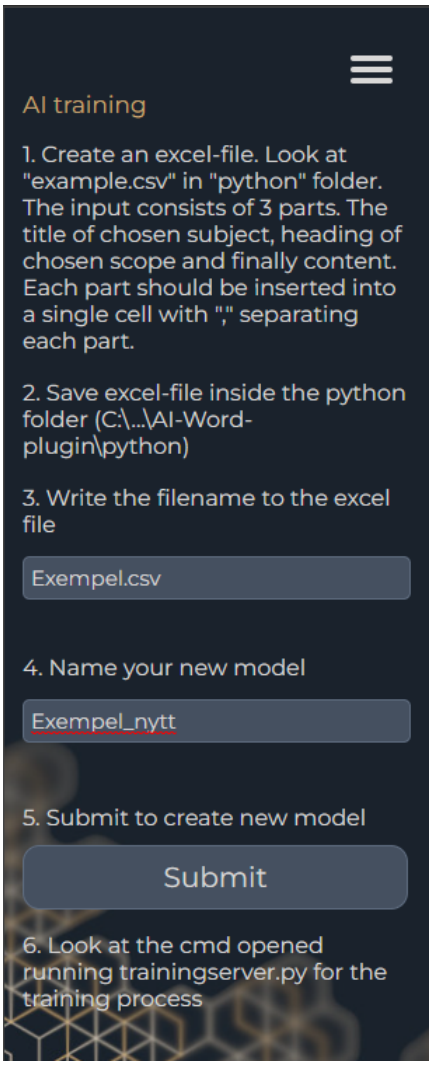
4.4 Tillgängliggörandet av plugin

Eftersom plugin-programmet inte är ämnat att publiceras och laddas upp online till Words Addin-bibliotek behövdes plugin-programmet tillgängliggöras på ett annat

sätt. Lösningen blev att skapa en exekverbar fil skriven i Python som startar både Word, Node.js samt en Python-server för AI-träning. För att installera programmet som en ny användare behöver användaren hämta filen med hela programmet och installera Python samt nödvändiga paket till Python. Ett planerat moment var att ladda upp plugin-programmet på en Microsoft Azure server, en molnplattform som kan agera ”värd” åt ett program. Detta skulle vara ett stort steg framåt i användarvänlighet av plugin-programmet och användaren hade inte haft tillgång till koden utan bara behövt en liten XML-fil. Detta bortprioriterades dock då det bedömdes ta för mycket tid.

4.5 Träning av AI

Ett av önskade målen från PO var att det skulle finnas möjlighet att träna en AI som sedan skulle kunna användas inuti Word. Tanken var ett den skulle kunna skriva ut texter likt hur någon specifik organisation skriver, eller skriva ut ny information den har tränats på. Detta var intressant eftersom det skulle få plugin-programmet att stå ut från liknande plugins eller AI-hjälpmedel. Fokuset på AI-träningen var att skapa en Q&A AI-modell som kunde svara på frågor om ett företag genom att läsa texter om det.

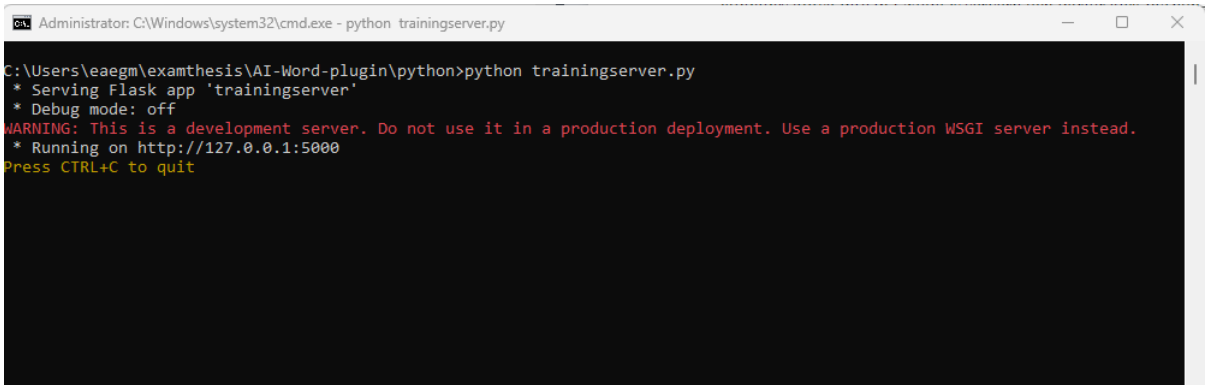


Figur 4.12: AI training

4.5.1 AI träning från Excel-dokument

Lösningen som skapades för att träna en AI är att användaren skapar en .csv fil med kolumnerna titel, rubrik och innehåll. Användaren fyller sedan i exempelvis: ”Historia”, ”Vikingatiden”, ”Vikingatiden varade från ca 700-talet”. Denna data används sedan av en existerande AI-modell från OpenAI som läser igenom och skapar frågor utifrån texten. Frågorna besvaras sedan med hjälp av samma AI utifrån den texten som matats in. Frågorna och svaren formateras så att de blir mer lika varandra, exempelvis avslutar varje fråga och svar med ett mellanslag. Detta gör att de skiljer sig åt mindre och att det då blir större fokus på innehållet i texten istället för hur den är formaterad. Sedan skrivs frågorna och svaren till en JSONL-fil där frågorna blir det som användaren skriver in och svaren är det som AI-modellen förväntas svara vid en sådan fråga. JSONL-filen skickas sedan till OpenAI som skapar en ny modell baserad på en davinci-003 modell och den data den tränats med. Modellen som är skapad är kopplad till kontot och kan laddas i plugin-programmet genom att skriva en API-nyckel från det kontot. I figur 4.12 visas hur AI-tränings sidan

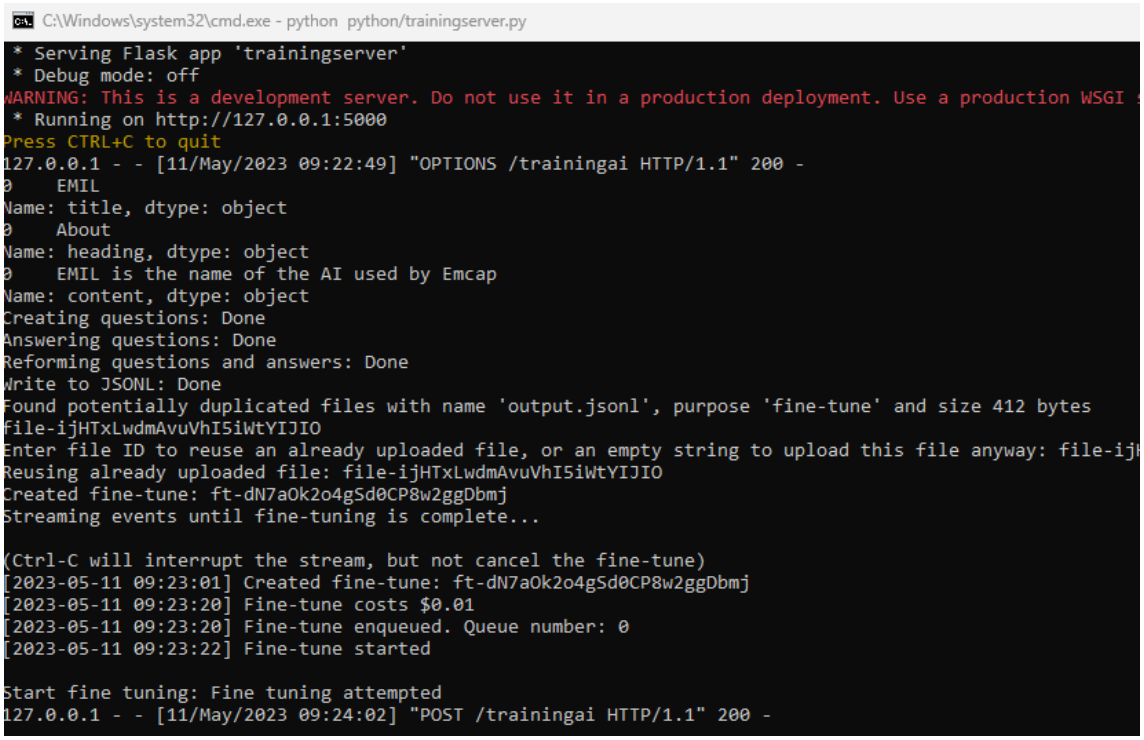
visas i plugin-programmet. För att träna en AI behöver användaren alltså skapa ett excel-dokument, spara på angiven plats, skriva in namnet på filen, namnge den nya modellen och klicka på knappen ”submit”.



Figur 4.13: Python Trainingserver i vänteläge

4.5.2 Python/HTTP-lösning

Eftersom AI-träning valdes att göra med hjälp av språket Python behövdes ett Python-program kunna köras från plugin-programmet. För att kunna köra Python-programmet skapades en kommunikationskanal mellan plugin och Python via HTTP. Vid uppstart av systemet kan ”trainingmode” aktiveras vilket gör att en lokal server körs där Python-programmet ligger, beredd att svara på anrop från plugin-programmet. Python-programmet använder sig av Flask för att bli en egen server samt CORS för att möjliggöra kommunikation mellan script från annat domän. Det CORS gör är att tillåta kommunikation mellan Javascript-filen på Node.js-servern och Python-programmet på Python-serven. Vid AI-träning görs ett ”POST” anrop via HTTP. När detta sker skickas namn på excelfil med data, namn på ny modell och API-nyckeln. Denna data är paketerad i en JSON-fil. Python-servern paketerar upp JSON-filen och använder data för att skapa en JSONL-fil med frågor och svar som skickas till OpenAI:s server, där en AI-modell tränas. I Figur 4.13 visas Python-servern när den är igång och väntar på anrop. I Figur 4.14 visas när Python-servern har tagit emot förfrågan, skapat frågorna, svarat på frågorna och skapat en ny tränad modell.



Figur 4.14: Python-server respons

5
Resultat

I detta kapitel presenteras resultatet från de olika målen som sattes och hur väl de uppfylldes. Varje delkapitel tar upp olika kategorier av målen för att de olika aspekterna av arbetet ska jämföras för sig.

5.1 Textgenerering

Det första målet med arbetet var att texten som genererades av OpenAI skulle skrivas ut direkt i ett Word-dokument. Detta mål uppfylldes tidigt i projektet genom att ha en liten textruta som tog in text och skickade den till OpenAI:s API som sedan returnerade ett svar baserat på texten. Målet att den skulle kunna skriva en text med rubrik och underrubriker med tillhörande textkroppar var även möjligt ifall man skrev in korrekta instruktioner. Korrekta instruktioner innebär i detta fall att användaren skriver in att rubriker och underrubriker ska vara med. Detta behövdes göras eftersom AI:n annars inte vet vilket format användaren önskar. Fler funktioner lades till i form av temperature och max_tokens för att låta användaren ha mer kontroll över det som skrivs ut.

5.2 Användarvänlighet

För att kunna besvara frågeställningarna ifall plugin-programmet kommer vara lätthanterligt och praktiskt med syfte att skriva ut en textmall, så skapades en enkät som utfördes av tre personer som inte hade varit involverade i projektet. De kunde då ge respons på hur plugin-programmet uppfattades ur en annan synvinkel. Personerna som valdes var familjemedlemmar och klasskamrater som var lätta att kontakta och som aldrig hade sett plugin-programmet tidigare. Enkäten byggdes upp på ett sätt där de tre första delarna hade uppgifter som skulle utföras innan användaren fick använda programmet fritt. Den fjärde delen skulle besvaras efteråt. De tre första delarna hade därför fokus på hur en person som bara sett plugin-programmet uppfattar det, medan den fjärde delen fokuserar mer på hur användaren upplevde programmet efter användning.

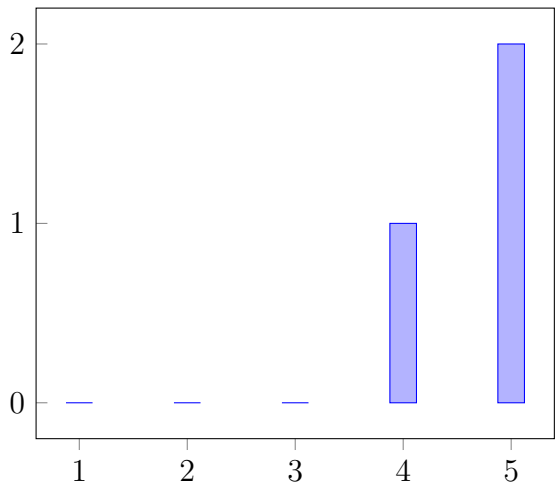
Den första delen av frågor bestod endast av en fråga och var "Vad är ditt första intryck av plugin-programmet när du öppnar programmet?". Denna fråga ställdes i syftet att ta reda på vad någon som ser den för första gången tänker. Denna fråga var intressant eftersom en del arbete hade spenderats på att se till att den inte såg

för avancerad ut för nya användare. Svaren på denna enkät presenteras i tabell 5.1.

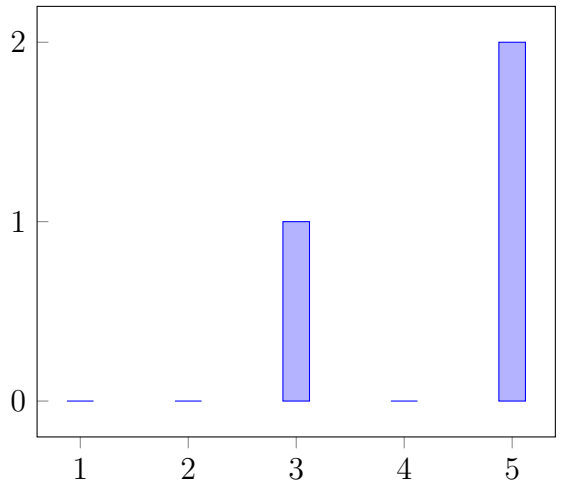
Användare	Svar
1	spännande, känns intressant
2	ser snyggt ut och passar optisk bra in i word
3	hm, hur ska man göra nu...

Tabell 5.1: Svar på frågan: ”Vad är ditt första intryck av plugin-programmet när du öppnar det?”

Den andra delen bestod av två frågor som rörde det visuella. Dessa frågor var en fortsättning på den första delen men var mer uppdelade i hur olika delar uppfattades, ifall det inte täcktes av svaret på första frågan. Frågorna berörde läsbarheten och färgvalet av plugin-programmet eftersom det hade spenderats en del arbete på dessa delar under designutvecklingen. Svaren på frågorna presenteras i figurerna 5.1 och 5.2.

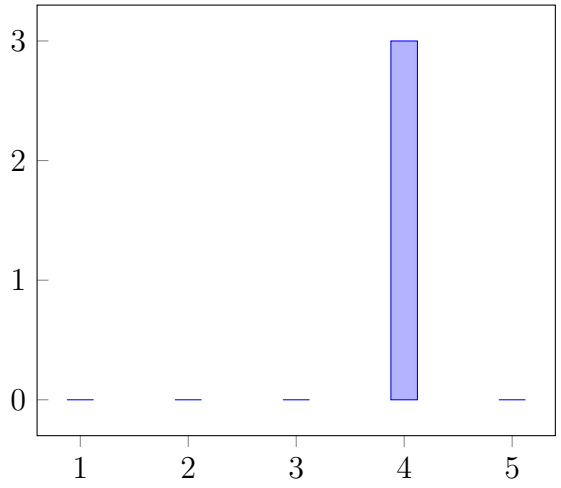


Figur 5.1: Svar på frågan: ”Hur enkelt/svårt tycker du det är att läsa rubrikerna till funktionerna?”. 1 är väldigt svårt och 5 är väldigt enkelt.

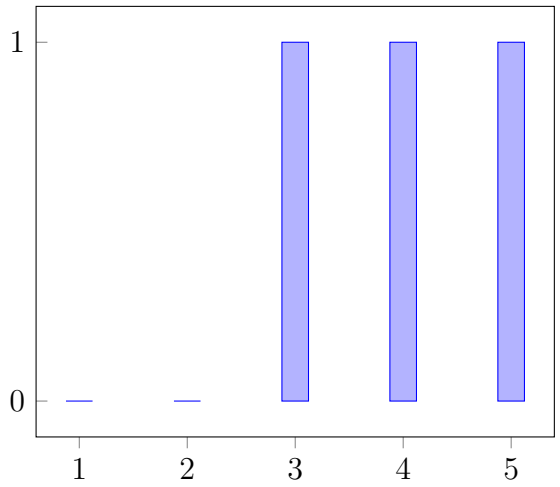


Figur 5.2: Svar på frågan: ”Hur väl passar färgerna ihop i designen?”. 1 är väldigt dåligt och 5 är väldigt bra.

Den tredje delen bestod av två frågor som berörde texterna som förklarar hur plugin-programmet används. Dessa frågor ställdes i syfte att ta reda på hur väl beskrivningarna förklarar vad de olika sakerna gör, för någon som inte redan vet vad de gör. Svaren kunde då indikera hur väl beskrivningarna hjälpte med att förhindra programmet från att bli allt för avancerat. Svaren på frågorna presenteras i figurerna 5.3 och 5.4.



Figur 5.3: Svar på frågan: ”Hur väl beskriver info-rutorna vad de olika funktionerna gör?”. 1 är väldigt dåligt och 5 är väldigt bra.



Figur 5.4: Svar på frågan: ”Hur väl beskriver placeholder texten i “Input” textrutan vad det valda läget gör?”. 1 är väldigt dåligt och 5 är väldigt bra.

Den sista delen bestod av sju frågor och berörde användarens upplevelse efter att de använt plugin-programmet. Dessa frågor ställdes för att ta reda på ifall det fanns aspekter som skulle kunna förbättras och för att ta reda på deras åsikter nu när de använt programmet. Svaren presenteras i tabellerna 5.2, 5.3, 5.4, 5.5 samt figurerna 5.5, 5.6 och 5.7.

Användare	Svar
1	
2	en gång i insert reagerade submit knappen inte.
3	Ja status på vad den håller på med

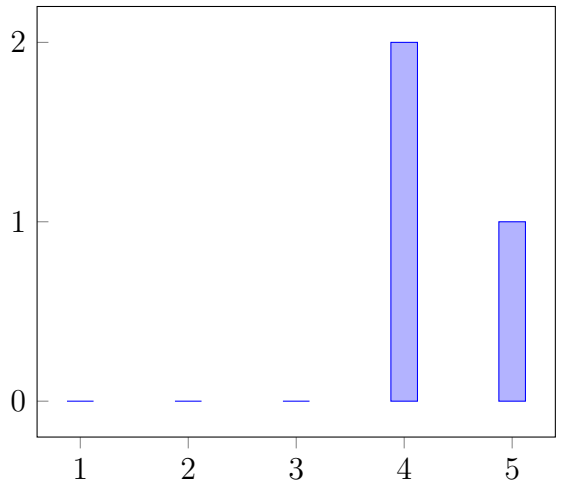
Tabell 5.2: Svar på frågan: ”Finner du något frustrerande med pluginet? I så fall vad?”

Användare	Svar
1	Justera ungefär hur mycket text som fylls i via [insert].
2	Light mode för lättare läsning ifall det reflekte ras ljus på skärmen
3	Att man tydligt ser när appen jobbar och när den är klar

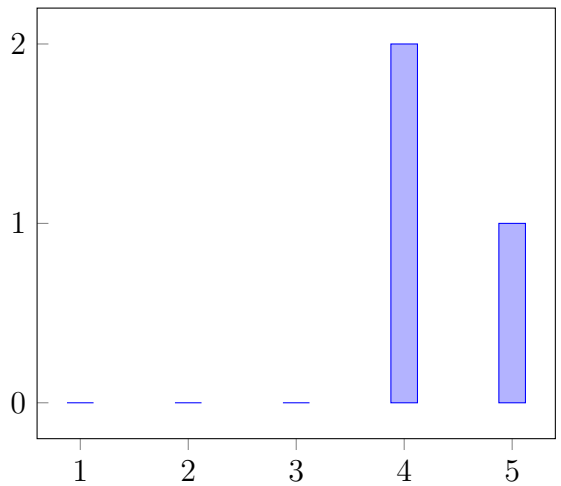
Tabell 5.3: Svar på frågan: ”Ifall du kunde förbättra en sak med appen, vad hade det varit?”

Användare	Svar
1	
2	nej
3	Generera musik

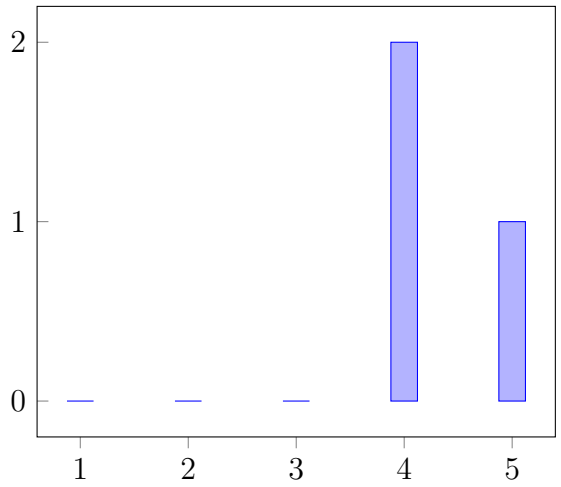
Tabell 5.4: Svar på frågan: ”Finns det någonting du önskar att appen kunde göra som den inte redan kan göra?”



Figur 5.5: Svar på frågan: ”Hur stor är sannolikheten att du skulle installera detta hjälp-plugin till ditt Word?”. 1 är inte alls troligt och 5 är väldigt troligt.



Figur 5.6: Svar på frågan: ”Hur enkelt/svårt tycker du att det är att använda vår applikation?”. 1 är väldigt svårt och 5 är väldigt enkelt.



Figur 5.7: Svar på frågan: ”Hur väl känner du att du vet hur man använder plugin-programmet nu?”. 1 är väldigt dåligt och 5 är väldigt bra.

Användare	Svar
1	
2	bra applikation
3	Lite mer hjälptexter och exempel skulle vara bra

Tabell 5.5: Svar på ”Övriga åsikter”

5.3 Träning av AI

Efter att träning utförts på en OpenAI-modell så skapas en ny modell som ska kunna ge svar på frågor som den har tränat på [25]. Den tränade modellen som skapades under detta arbete lyckades däremot inte skriva särskilt bra resultat. Allting som den skrev ut repeterades många gånger, vissa av gångerna togs frågan som ställdes med i utskriften medan andra gånger så skrevs bara svaret ut. Vid träning fick modellen vissa värden som den skulle svara på vid vissa frågor, men när modellen sedan testades svarade den ofta med andra värden.

5.3.1 Träning på Volvos årsredovisning 2022

Ett försök som gjordes var att träna på Volvos årsredovisning 2022. Denna information valdes eftersom den var nyutgiven vid tillfället och för att säkerställa att informationen inte redan fanns hos hos någon av Open AI:s modeller. 30 stycken rader med information skapades i ett excel-dokument som en AI-modell fick träna på. En av raderna löd exempelvis: ”Volvo Group Annual Report 2022, Earnings per share of SEK 16.09”. När den tränade modellen sedan fick frågan ”Hur stor förtjänst per aktie gav Volvo Group år 2022?”, svarade den att förtjänsten per aktie var 15.0 SEK, samt repeterade svaret 20-30 gånger. Modellen verkar alltså ha avrundat svaret neråt, vilket inte ger ett helt korrekt svar.

6
Diskussion

I detta kapitel diskuteras resultaten som presenterats. Först diskuteras projektet utifrån frågeställningarna introducerade i inledningen och ifall de går att besvaras. Sedan diskuteras möjligheten att vidareutveckla arbetet.

6.1 Användarvänligheten enligt enkäten

Projektet gick snabbt framåt i början, med enbart något litet hinder vid användningen av OpenAI:s API. Detta gjorde att mycket fokus kunde spenderas på användarvänlighet och ett möte med designutvecklaren på Emcap gav tidigt värdefull feedback. Detta gjorde att designen hade hunnit gå igenom mycket planerande innan enkäten väl utfördes. Enkäten utfördes däremot bara av tre människor vilket ökar chansen för att urvalsfel kan uppstå. Detta innebär att ifall något av svaren innehåller statistiskt avvikande värden så kommer de få en mycket större påverkan på resultatet än vad det hade fått vid en större mätning. Resultaten från enkäten kan därför vara dålig på att återspegla hur plugin-programmet skulle uppfattas av ett större urval av människor.

Resultaten som kom från enkäten var för det mesta positiva. Designen fick bra respons förutom ett svar som visas i tabell 5.1 som kan tolkas som att en användare var förvirrad vid det första intrycket. Detta var inte optimalt eftersom plugin-programmet designades så att den inte skulle vara för avancerad vid det första intrycket. Detta svar var det som stod ut mest från hela enkäten. Eftersom det var få svar är det dock svårt att se ifall det är en vanlig eller ovanlig åsikt.

På den andra delen som mer rörde det visuella, var åsikterna mestadels positiva. Respondenternas svar visas i figur 5.1 samt 5.2. Svaren visar att de fann rubrikerna lättlästa och att färgerna i designen passade ihop för det mesta. Även detta var någonting som hade spenderats en del arbete på, bland annat genom att göra ett motiv i bakgrunden suddig för att göra rubrikerna mer lättlästa. Det var därför positivt att se att respondenterna tyckte att designvalet var bra. På frågan kring färgvalet fanns det en respondent som endast gav godkänt betyg, medan de andra fann att färgerna passade ihop väldigt bra. Färgerna som kunde väljas var utifrån Emcaps grafiska profil och var därför begränsade, men eftersom majoriteten fann att de passade så tolkas det som ett bra val.

Den tredje delen, som rörde beskrivningar, hade lite sämre respons än de tidigare

delarna. Svaren från figur 5.3 visar att info-rutorna beskrev de olika funktionerna hyfsat bra, men det var ingen som gav den full poäng. Eftersom ett av målen var att plugin-programmet skulle vara lätthanterligt så visar denna respons på att denna del hade kunnat utvecklas lite mer. Det var ingen som fann beskrivningarna så pass otydligt att dom gav någon poäng under tre, så därför ses detta som tillräckligt bra för plugin-programmets användarvänlighet. Figur 5.4 hade liknande svar och placeholder-texten skulle möjligtvis kunna förbättras något. Texten som användes till placeholder-texterna togs av OpenAIs beskrivningar eftersom dessa ansågs vara tydliga, men efter responsen så är det möjligt att texten skulle kunna förtydligas ytterligare.

Den sista delen utfördes efter att användarna hade använt plugin-programmet och innehöll en del givande förslag till förbättringar. Ett av dessa förslag visas i tabell 5.3 och var att ha ett ljust läge där färgen på plugin-programmet blir ljusare, för att det ska vara lättare att läsa när det finns reflexioner på skärmen från omgivningen. I början av projektet så var designen väldigt ljus och det fanns då ett förslag på att skapa ett mörkt läge som skulle vara bekvämligare att läsa när det är mörkt. När designen sedan gjordes om till en mörk design så lades idén om två lägen på is eftersom detta hade lägre prioritet och sågs som en extrafunktion. Två olika lägen hade varit mycket möjligt att implementera ifall det hade funnits mer tid. Ett ytterligare förslag som visas i tabell 5.5 är att det hade kunnat finnas mer hjälptexter. Detta hade inte heller varit något som varit svårt att implementera och borde även gjort den mer lätthanterlig för människor som inte använt plugin-programmet tidigare. Ett annat förslag som visas i tabell 5.4 och 5.5 var att implementera något grafiskt som visar när plugin-programmet arbetar så att användaren vet att det som skickade in till programmet, håller på att arbetas på. Detta hade också varit möjligt att implementera och förmodligen lett till mindre frustration för användaren, eftersom användaren i det fallet vet om knappen submit har blivit klickad på eller inte. Detta var något som uppstod för användare nummer 2 då det var en fördröjning när personen hade klickat på submit. När ingen respons gavs av plugin-programmet visste inte användaren om knappen hade registrerat inmatningen eller inte.

Det kom även två andra förslag på ändringar i tabell 5.4 och tabell 5.5 vilka var att justera hur mycket text som fylls ut via [insert] och att kunna generera musik. Att kunna justera hur mycket text som [insert] skriver är något som delvis är möjligt ifall man ger instruktioner på exempelvis hur många meningar AI:n ska fylla ut med. Det är däremot svårt för den att fylla innehållet med mycket lite text, som exempelvis endast med ett ord. Det är möjligt att detta skulle kunna förbättras med AI-träning men det är något användaren skulle behöva göra själv i sådana fall. Musikgenerering är någonting som aldrig kom på tanken under projektet och mycket av det ligger i att OpenAIs API inte kan svara med ljudfiler för tillfället. Detta blir därför svårt att uppfylla med de AI-modeller som användes i projektet. För att uppfylla det skulle andra AI modeller behöva användas. Det närmsta man kan komma detta önskemål för tillfället är att generera en låttext och sedan använda den inbyggda funktionen i Word för att läsa upp texten.

Frågorna som visas i figur 5.5, 5.6 och 5.7 berör användarnas åsikter kring deras totala upplevelse med plugin-programmet. Svaren hade ett snitt på 4,3 av 5 där högre poäng är positivt och visade att användarna fann plugin-programmet enkelt att använda och att användarna hade fått en bra förståelse över hur man använder programmet. Enkäten visar att, några människor som inte använt plugin-programmet tidigare, kände att de under en kort tid lärde sig använda programmet väl och att det var enkelt att använda. Från detta kan man dra slutsatsen att plugin-programmet var lätthanterligt.

Möjligheten att skriva ut en textmall via plugin-programmet möjliggjordes tidigt under projektet, men kvaliteten på textmallen beror till stor del på vad användaren skriver för instruktioner. Textgenerering blir mer användbar när användaren har mer erfarenhet inom instruktionsskrivning och har en bättre förståelse över hur plugin-programmet fungerar. Enkäten visade att användarna fann programmet lätt att använda och att de kände att de fick en bra förståelse över hur det fungerar. Från detta kan man dra slutsatsen att programmet är tillräckligt välutvecklat för att en användare ska kunna skriva ut en användbar textmall.

6.2 Diskussion kring AI-träning

AI-träningsmomentet gick däremot inte lika bra och tog mer tid än planerat. Orsaken till detta var att det uppstod problem som saktade ner utvecklingen. Bland annat hur man exekverar ett Python-script från Node.js. Därför kunde endast begränsad tid spenderas på att förbereda data för träning och justering av parametrar som hade kunnat leda till bättre resultat. Exempelvis gav OpenAI rådet att ha omkring 500 rader med information för en lyckad AI-träning, detta kan jämföras med de 30 rader som användes i testet för Volvos årsredovisning 2022. Med den begränsade tiden som spenderades på träning är det därför svårt att dra en generell slutsats ifall träning av OpenAI-modeller kan ge ett önskat utfall. Plugin-programmet lyckades inte träna en modell som skrev ut korrekta svar. Däremot lyckades programmet med träning på data och att skapa en egen modell som kunde skriva ut svar relaterad till informationen den blev matad med. Slutsatsen är att AI-träning är möjlig och att med mer tid för utveckling hade bättre resultat kunnat uppnås. Att kunna använda den tränade AI:n för att göra anpassade mail, rapporter och så vidare hade alltså säkert varit möjligt om större träningsmallar hade använts.

6.3 Styrkor och svagheter med valda metoder

I denna del följer diskussion om valda tillvägagångsätt.

6.3.1 Planering

Planeringen gjordes de två första veckorna av projektet och kan ses i appendix A.2. Eftersom projektetmetoden var Scrum var denna planering bara en approximation.

Datum och tidsåtgången för olika sprints var därför möjliga att ändra i efterhand, vilket även gjordes i ett tidigt skede. Tidsåtgången för att få till ett ”Hello World” visade sig vara kraftigt överskattat varpå tidsschemat gjordes lite mer ambitiöst. I det stora fungerade tidsplanen bra. Målen som hade blivit uppsatta följdes och gjorde att projektet framskred i ett kontinuerligt tempo.

6.3.2 Scrum

Att använda Scrum var användbart eftersom projektet både snabbades på vid vissa tillfällen men även drog ut på tiden vid andra. En del funktioner och prioriteringar tillkom även genom önskemål från Emcap, eller efter att vissa möjligheter upptäcktes, vilket gjorde att det agila tillvägagångssättet blev mycket användbar. Eftersom ingen av författarna hade någon tidigare erfarenhet av Node.js, Word Add-ins eller Open AIs API hade det varit svårt att göra projektet enligt en mer klassisk vattenfallsmodell. För mjukvaruutveckling överlag är det svårt att förutse svårigheter och begränsningar som kan uppstå under utvecklingens gång, vilket talar för att den agila arbetsmetoden är lämplig.

6.3.3 Angreppsätt

Arbetet utgick från målen som var uppsatta enligt GANTT-schemat, se Appendix A.2. Angreppsättet var att försöka lösa uppgiften samtidigt som kunskaper samlades in via guider på internet och andra källor. Denna metod fungerade bra då båda författarna anser att man bäst lär sig när en uppgift behöver lösas, i kontrast till att först efterforska försöka få en bred kunskapsbas, innan man angriper problemet. Ibland fastnade båda utvecklarna på olika saker där det var svårt för båda att dyka in tillräckligt djupt i varandras problem för att hjälpa varandra. Eftersom arbetet bara utfördes av två personer är detta problem dock svårt att åtgärda. I efterhand ser vi återvändsgränder som uppstod och som inte hade uppstått med mer erfarenhet eller med fler utvecklare som var djupare insatta i projektet. Exempelvis när projektet fastnade på Python-implementeringen av AI-träning. Båda utvecklarna fick uppfattningen från OpenAI:s hemsida att enbart Python kunde användas för att träna en AI. Med lite efterforskning hade det kunnat hittas att OpenAI även hade stöd för träning via Node.js. Den informationen hade sparat mycket tid.

6.4 Etik

Nedan följer några etiska frågeställningar samt möjliga konsekvenser av tekniken.

6.4.1 Etiska aspekter kring AI

En AI baserad textgenerator skapar texter utifrån vad AI:n har matats med. Det den returnerar är baserat på den data den hittar om ämnet och den kommer därför inte fram till nya slutsatser. En AI kan även bli matad med felaktig information vilket skulle kunna leda till att felaktiga slutsatser skapas. Med detta i hänseende finns det etiska aspekter kring denna teknik eftersom studenter, personal, konsulter

osv. kan inspireras och hämta slutsatser från en AI istället för att hämta fakta från vetenskapliga källor. Om tekniken används på det sättet kan felaktiga slutsatser dras. För studenter, där syftet är att bearbeta tankar och information genom att skriva, skulle resultatet av att använda AI riskera att bli att kunskaper och en utvecklad förmåga uteblir. Möjligtvis skulle även gemene användares kreativa förmåga minska när det gäller textskrivning eftersom mindre kreativitet behöver användas för att skapa texten.

Ett annat problem med användandet av AI är att den data som AI:n läser av inte är neutral, utan är viktad åt olika håll [26]. Företag som äger populära AI skulle därför kunna påverka allmänheten i olika riktningar genom att bestämma vilka svar som är önskade från AI:n. AI-modeller kan även bli påverkade om personer hittar brister eller på annat sätt påverkar den data AI:n tränar på. Detta kan leda till allvarliga problem. En AI kan exempelvis lära sig rasistiska fraser och även diskriminera människor baserat på hudfärg, kön etc. Ett exempel är företaget Amazon som hade ett AI-program som användes för att hitta rekryter till deras tekniska avdelningar, men som diskriminerade kvinnor [26]. Microsoft lanserade även en chattbot som drevs med hjälp av AI. Efter 24 timmar i bruk hade denna chattbot lärt sig rasistiska fraser av användare som den sedan använde i sin kommunikation [26].

6.4.2 Samhälleliga aspekter

En möjlig framtida konsekvens av denna teknik kan vara att frågeställningar och uppgifter som ges till studenter förändras. Uppgifter skulle alltså kunna bli mer specifika från start för att undkomma enkel "AI-plagiering". Om exempelvis en student idag får till uppgift att skriva en text om hur andra världskriget gick till kanske samma student i framtiden får i uppgift att beskriva hur en fiktiv person upplevde en dag under andra världskriget och där den fiktiva personen har några fördefinierade egenskaper och relationer. Detta för att säkerställa att texten är autentisk och för att visa att studenten själv har bearbetat texten.
I övrigt skulle textskapandet kunna förändras i samhället. Mindre tid kanske läggs på att skriva texter och större fokus går åt till att hitta "rätt slags frågor". En annan möjlig effekt skulle kunna vara att källkritik får en större roll. Eftersom texter blir ihopskrivna av AI-generatorer där felaktig information kan vara inlagd, skulle effekten av detta kunna bli att användare blir mer kritiska till innehållet i texter.

6.4.3 Ekologiska aspekter

Under projektet utvecklas bara mjukvara och påverkan på miljön blir därmed den energi som krävs för att datorer och servrar ska vara i gång samt slitage på dessa som sedan kan behöva ersättas.
För att skapa AI-modeller krävs mycket träning och data vilket använder mycket energi. Det är möjligt att den totala mängden energi som förbrukas vid användning av AI är större än vad användaren hade använt ifall texten hade skrivits på ett konventionellt sätt. Däremot är det också möjligt att det går snabbare att få svar

från AI:n än ifall användaren själv skulle söka informationen. Det är därför både möjligt att användaren skulle kunna förbruka mer eller mindre energi genom att använda AI än ifall de inte skulle göra det.

6.5 Fortsatt utveckling

Under arbetets gång publicerades det ett plugin av Smart Barn Technologies som kunde utföra samma sak som detta projekts MVP var planerad att kunna utföra [27]. Den fördel detta projekt har gentemot Smart Barns plugin är potentialen att kunna träna en AI-modell. Detta arbete är dock ett koncept för att utforska ifall en AI-driven textgenererande- Word-plugin skulle vara möjlig att utveckla och planeras därför inte att byggas vidare på i framtiden. Träningsmomentet gav inte önskade resultaten och krånglade till mycket av programkoden. Träningen är hårdkodad för att bara kunna utföra Q&A träning så om man hade önskat en annan slags träning hade koden behövt skrivas om. För tillfället fungerar den bara om man kör den lokalt och ifall man hade vidareutvecklat konceptet hade ett steg förmodligen varit att ladda upp det på en server. Detta moment hade blivit mycket krångligare på grund av den nuvarande träningsimplementationen och det lättaste hade varit att skriva om stora delar av träningskoden eller implementera en separat applikation som bara fokuserar på att träna AI-modeller istället. Det upptäcktes även efter att utvecklingsmomentet av arbetet var utfört att det hade varit möjligt att utföra träning utan att använda sig av Python. Ifall man då hade skrivit om programmet hade man möjligtvis kunnat göra det med Javascript vilket skulle göra det lättare att använda.

7
Slutsats

Detta projekt har undersökt implementering av AI i egen mjukvara. Ett nytt unikt koncept har skapats. Projektet har utforskat nya idéer för hur AI kan användas i praktiken. Den 16:e Mars 2023, precis två månader efter att detta projekt startades, lanserade Microsoft sin egen variant: Copilot 365 [28]. AI Word Plugin är inte menat att konkurrera med Microsoft, däremot att undersöka, testa och hitta nya idéer. Utvecklingen pekar mot att hjälpmedel som dessa kommer vara standard i framtiden. AI har stor potential att förenkla våra liv. Samtidigt är utvecklingen av AI resurskrävande i både datakraft och energi. Stora företag kommer vara drivande i utvecklingen. För att träna en egen AI-modell behöver man antingen skapa en egen, vilket skulle ta mycket stora resurser, eller använda en befintlig vilket i sig kan bli en stor kostnad för att betala avgifter för att få använda tjänsterna. Då är även valmöjligheter och kontroll av AI:n begränsade. Detta projekt kan vara ett exempel på hur liknande koncept kan utvecklas i framtiden och bana väg för fortsatt utveckling.

Litteratur

[1] T. Hagendorff and K. Wezel. "15 challenges for AI: or what AI (currently) can't do", *Springer Link*. Mar, 28, 2019. Hämtad: 2023-05-12. [Online.] Tillgänglig: https://link.springer.com/article/10.1007/s00146-019-00886-y

[2] B.Bergstein. "What AI still can't do", *MIT Technology Review*, Feb, 19, 2020. [Online]. Tillgänglig: https://www.technologyreview.com/2020/02/19/868178/what-ai-still-cant-do/, Hämtad: 2023-05-12.

[3] A.Ng. "What Artificial Intelligence Can and Can't Do Right Now", *Harvard Business Review*, Nov, 19, 2016. [Online]. Tillgänglig: http://www.w-t-w.org/de/wp-content/uploads/2016/11/Andrew-Ng-What-AI-Can-and-Can%E2%80%99t-Do.pdf, Hämtad: 2023-05-12.

[4] P.Lee. "Learning from Tay's introduction", *microsoft*, Mar, 25, 2016. [Online]. Tillgänglig: https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/, Hämtad: 2023-06-06.

[5] *AI-writer*, "AI-writer". [Online]. Tillgänlig: https://ai-writer.com/, Hämtad: 2023-05-12.

[6] *Simplified*, "Simplified". [Online] Tillgänlig: https://simplified.com/ai-writer/, Hämtad: 2023-05-12.

[7] *OpenAI*, "Introducing ChatGPT', Nov, 30, 2022. [Online]. Tillgänglig: https://openai.com/blog/chatgpt#OpenAI, Hämtad: 2023-05-12.

[8] K.Jeremy, L.Michal and M.Jessica "ChatGPT creates an A.I. Frenzy", in *Fortune* Volym 187 nummer 1 sidor 44-53 in 2023. [Online]. Tillgänglig: https://search.ebscohost.com/login.aspx?direct=true&db=bsu&AN=161458918&site=eds-live&scope=site, Hämtad: 2023-05-12.

[9] *OfficeDev*, "Yeoman generator for Office Add-ins - YO OFFICE!". [Online]. Tillgänglig: https://github.com/OfficeDev/generator-office, Hämtad: 2023-05-03.

[10] *Figma*, "Figma: the collaborative interface design tool". [Online]. Tillgänglig: https://www.figma.com/, Hämtad: 2023-05-12.

[11] *GitHub*, "GitHub: Let's build from here". [Online]. Tillgänglig: https://github.com/, Hämtad: 2023-05-12.

[12] *W3C*, "HTML & CSS". [Online]. Tillgänglig: https://www.w3.org/standards/webdesign/htmlcss, Hämtad: 2023-05-12.

[13] R.Ferguson. "Beginning JavaScript" 3rd ed. Ocean, NJ, USA: Apress 2019

[14] *Node.js*, "About Node.js®". [Online]. Tillgänglig: https://nodejs.org/en/about/, Hämtad: 2023-05-12.

[15] *Webpack*, "Webpack". [Online]. Tillgänglig: https://webpack.js.org/, Hämtad: 2023-05-9.

[16] *npm*, "About npm". [Online]. Tillgänglig: https://docs.npmjs.com/about-npm, Hämtad: 2023-05-12.

[17] M. Lutz, *Learning Python*, 4th ed., Ed., Julie Steele, Sebastopol, U.S.A., O'Reilly, 2009. kap. 1, ss 3-5 [Ebook] Tillgänglig: https://books.google.se/books?hl=en&lr=&id=ePyeNz2Eoy8C&oi=fnd&pg=PP1&dq=Learning+Python+mark+lutz&ots=McBabH6cAl&sig=HATdEoNrAcZr4fZ72DRl1IM-1Ck&redir_esc=y#v=onepage&q&f=false

[18] J.F. Kurose, K.W. Ross, *Computer Networking A Top-down Approach Global ed.*, 8th ed., Pearson, Harlow, UK, 2022

[19] *npm*, "OpenAI Node.js Library". [Online]. Tillgänglig: https://www.npmjs.com/package/openai, Hämtad: 2023-05-04.

[20] *Pallets*, "Flask". [Online]. Tillgänglig: https://github.com/pallets/flask, Hämtad: 2023-05-04.

[21] *IETF*, "The Web Origin Concept". Dec, 2011. [Online], kap. 3 Tillgänglig: https://datatracker.ietf.org/doc/html/rfc6454#page-4, Hämtad: 2023-05-09.

[22] R. Gunasundaram, R. Goya, "A Python Q&A Session" i *CORS Essentials*, Birmingham, UK, Packt, 2017. kap. 1, ss 1 [Ebook] Tillgänglig: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=4868541&query=CORS+Essentials, Hämtad: 2023-05-09.

[23] I.Ward, "JSON Lines", *JSON Lines*. [Online]. Tillgänglig: https://jsonlines.org/, Hämtad: 2023-05-09.

[24] *Pandas*, "Intro to pandas". [Online]. Tillgänglig: https://pandas.pydata.org/docs/getting_started/index.html, Hämtad: 2023-05-09.

[25] *OpenAI*, "Fine-tuning". [Online]. Tillgänglig: https://platform.openai.com/docs/guides/fine-tuning, Hämtad: 2023-05-04.

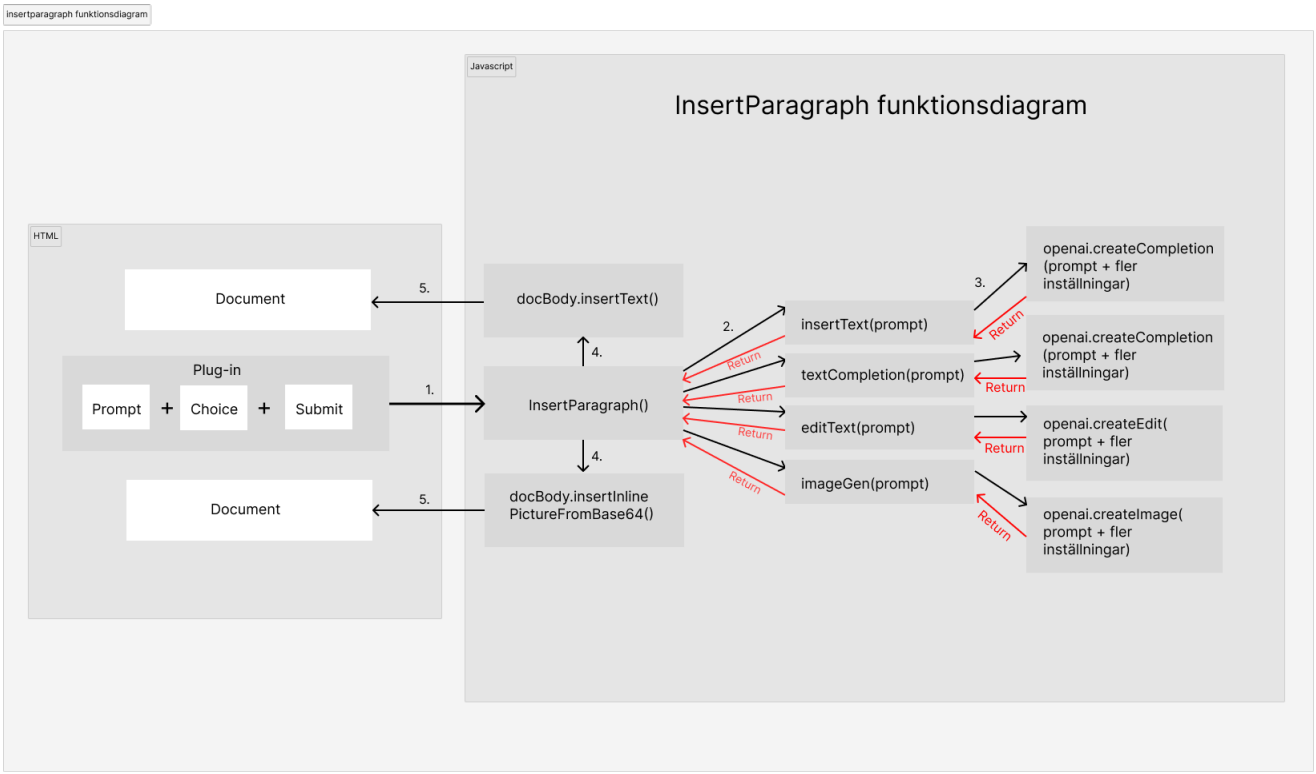
[26] Soraya Cardenas, Serafin F. Vallejo-Cardenas, "Continuing the Conversation on How Structural Racial and Ethnic Inequalities Affect AI Biases", 2019. Section 1, [Online] Tillgänglig: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=4868541&query=CORS+Essentials, Hämtad: 2023-06-01.

[27] *Pure info tech*, "Microsoft Word gets ChatGPT AI support with new add-in". [Online]. Tillgänglig: https://pureinfotech.com/microsoft-word-chatgpt-ai-addin/, Hämtad: 2023-05-11.

[28] *Official Microsoft Blog*, "Introducing Microsoft 365 Copilot – your copilot for work". [Online]. Tillgänglig: https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/, Hämtad: 2023-06-10.

A

Appendix 1



Figur A.1: Insertparagraph() - Fler funktioner

GANTT-schema

AI-wordplugin

#	Aktivitet	Beroende	Sprints								
			Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9
			30/1-10/2	13/2-24/2	27/2-10/3	13/3-24/3	27/3-7/4	10/4-21/4	24/4-5/5	8/5-19/5	22/5-2/6
A1	GUI										
A2	Basic "Hello World" ADD-in										
A3	Communication with AI										
A4	Writing inputs in the Add-in and	A1									
A5	Inställningar med AI	A4									
A6	GUI med fler val	A1-4									
B1	First produkt, test, dokument, tweeks	A1-6									
B2	Förbered användarenkät										
M1	1st MVP klar	A1-5				Deadline					
B3	Ytterligare funktioner	M1									
C1	Undersök att mata AI med data	A-B									
C2	AI skriver ut anpassat till hemsida	C1									
D7	Final buggtestning, finskrivning kod	A-C									
D8	Ytterligare utv. tid för rapport/present										
M2	Färdig produkt utefter POs önskan										Deadline
T1	Rapport										
T2	Presentation										

Figur A.2: Gantt-schema

INSTITUTIONEN FÖR Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS