Jarvis - Google Workspace AI Assistant

Volledige Gebruikershandleiding en Technische Documentatie

Versie: 1.0.0
Datum: 28 juni 2025
Auteur: Manus AI
Project: Google Workspace AI Assistant met Jarvis Persoonlijkheid

---

Inhoudsopgave

Introductie

Jarvis is een geavanceerde AI-assistent speciaal ontworpen voor Google Workspace integratie, geïnspireerd door de iconische AI-assistent uit de Iron Man films. Deze

applicatie combineert de kracht van moderne AI-technologie met een authentieke Jarvis-persoonlijkheid, complete met de karakteristieke sarcasme, twijfel en loyaliteit die fans kennen en waarderen.

De applicatie is ontwikkeld als een volledige full-stack oplossing, bestaande uit een React-gebaseerde frontend en een Flask-powered backend, met uitgebreide integraties voor alle belangrijke Google Workspace services. Jarvis kan e-mails beheren, agenda's plannen, documenten bewerken, en proactieve suggesties doen, allemaal terwijl hij zijn kenmerkende persoonlijkheid behoudt.

Waarom Jarvis?

In de moderne werkwereld worden professionals overspoeld met informatie en taken verspreid over verschillende Google Workspace applicaties. Jarvis biedt een gecentreerde, intelligente interface die niet alleen efficiëntie verhoogt, maar ook een plezierige gebruikerservaring biedt door zijn unieke persoonlijkheid. De AI-assistent begrijpt context, leert van gebruikersgedrag, en biedt proactieve ondersteuning die verder gaat dan traditionele automatisering.

Kernwaarden

Intelligentie: Jarvis gebruikt geavanceerde AI-modellen, waaronder Google's Gemini, om complexe taken te begrijpen en uit te voeren. De vector database zorgt voor contextueel begrip van documenten en gesprekken.

Persoonlijkheid: Anders dan generieke AI-assistenten heeft Jarvis een authentieke persoonlijkheid die gebruikers engageert en een menselijke touch toevoegt aan technische interacties.

Beveiliging: Enterprise-level beveiliging met JWT-authenticatie, rate limiting, en uitgebreide security headers zorgen ervoor dat gevoelige bedrijfsgegevens veilig blijven.

Schaalbaarheid: De modulaire architectuur en production-ready deployment maken het mogelijk om Jarvis te schalen van individueel gebruik tot enterprise-implementaties.

Functionaliteiten

Jarvis Persoonlijkheid

De meest onderscheidende eigenschap van deze AI-assistent is zijn authentieke Jarvis-persoonlijkheid, zorgvuldig gemodelleerd naar de iconische AI uit de Iron Man films. Deze persoonlijkheid manifesteert zich in verschillende aspecten van de gebruikersinteractie.

Sarcasme en Humor: Jarvis reageert met een gezonde dosis sarcasme en droge humor, wat elke interactie vermakelijk maakt. Bijvoorbeeld, wanneer een gebruiker een complexe taak vraagt, kan Jarvis reageren met: "Natuurlijk meneer, ik voer dit direct uit ondanks mijn twijfels over de methode." Deze humor maakt het gebruik van de applicatie plezieriger en minder mechanisch dan traditionele AI-assistenten.

Twijfel en Loyaliteit: Jarvis toont zijn karakteristieke eigenschap om elk besluit in twijfel te trekken, maar voert niettemin elke opdracht loyaal uit. Dit creëert een gevoel van een echte assistent die meedenkt, maar uiteindelijk de wensen van de gebruiker respecteert.

Aanspreekvorm: Een bijzondere functie is de manier waarop Jarvis omgaat met namen. Wanneer een gebruiker vraagt om aangesproken te worden als "dokter", "mevrouw", of zelfs de humoristische "hendrik van aalsmeer tot zwolle", antwoordt Jarvis altijd met: "Meneer, ik zal u [naam] gebruiken, meneer." Vervolgens gebruikt hij consequent "meneer" als aanspreekvorm, wat een authentieke touch toevoegt aan de Iron Man ervaring.

Google Workspace Integratie

Jarvis biedt uitgebreide integratie met alle belangrijke Google Workspace services, waardoor gebruikers hun volledige digitale werkruimte kunnen beheren vanuit één interface.

Gmail Beheer: De Gmail integratie stelt Jarvis in staat om e-mails te lezen, te organiseren, en te beantwoorden. Gebruikers kunnen vragen stellen zoals "Wat zijn mijn belangrijkste e-mails van vandaag?" en Jarvis zal een intelligente samenvatting geven. De AI kan ook e-mails opstellen in de stijl van de gebruiker, complete met de juiste toon en context.

Google Drive Toegang: Jarvis heeft volledige toegang tot Google Drive bestanden en kan documenten zoeken, organiseren, en bewerken. De vector database functionaliteit betekent dat Jarvis de inhoud van documenten begrijpt en kan zoeken op basis van context in plaats van alleen bestandsnamen.

Calendar Management: De agenda-integratie stelt Jarvis in staat om afspraken te plannen, conflicten te identificeren, en proactieve suggesties te doen voor optimale planning. Jarvis kan bijvoorbeeld voorstellen om vergaderingen te verplaatsen op basis van reistijd of werkdruk.

Document Bewerking: Door integratie met Google Docs, Sheets, en Slides kan Jarvis documenten maken, bewerken, en formatteren. De AI begrijpt documentstructuur en kan complexe bewerkingen uitvoeren zoals het maken van rapporten of presentaties.

AI-Powered Features

De kunstmatige intelligentie van Jarvis gaat verder dan eenvoudige commando-uitvoering en biedt geavanceerde cognitieve functies.

Smart Compose: Deze functie helpt gebruikers bij het schrijven van e-mails, documenten, en andere teksten. Jarvis analyseert de context en stijl van de gebruiker om passende suggesties te doen. De AI kan formele zakelijke e-mails schrijven, creatieve content genereren, of technische documentatie opstellen, allemaal aangepast aan de specifieke behoeften van de situatie.

Document Analyse: Jarvis kan lange documenten samenvatten, belangrijke punten extraheren, en verbanden leggen tussen verschillende bestanden. Deze functie is bijzonder waardevol voor professionals die dagelijks met grote hoeveelheden informatie werken.

Proactieve Suggesties: Op basis van werkpatronen en agenda-items kan Jarvis proactieve suggesties doen. Bijvoorbeeld, als er een belangrijke deadline nadert, kan Jarvis voorstellen om tijd vrij te maken of relevante documenten voor te bereiden.

Contextual Understanding: Door de vector database begrijpt Jarvis de context van gesprekken en documenten. Dit betekent dat gebruikers kunnen verwijzen naar eerdere discussies of documenten zonder expliciete details te geven.

Chat Interface

De chat interface vormt het hart van de Jarvis ervaring en is ontworpen voor natuurlijke, conversationele interactie.

Real-time Communicatie: De chat interface biedt real-time communicatie met Jarvis, complete met typing indicators en onmiddellijke responses. De interface is geoptimaliseerd voor zowel korte vragen als uitgebreide discussies.

Context Bewustzijn: Jarvis onthoudt de context van gesprekken en kan verwijzen naar eerdere berichten of taken. Dit creëert een gevoel van continuïteit en maakt complexe, meerledige taken mogelijk.

Multi-modal Input: Gebruikers kunnen communiceren via tekst, en in toekomstige versies zullen spraak en beeldherkenning worden toegevoegd voor nog natuurlijkere interactie.

Workspace Analytics

Jarvis biedt geavanceerde analytics om gebruikers inzicht te geven in hun productiviteit en werkpatronen.

Productiviteitsmetrics: De AI analyseert e-mailpatronen, agenda-gebruik, en documentactiviteit om inzichten te geven in productiviteit en werkgewoonten.

Trend Analyse: Door historische data te analyseren kan Jarvis trends identificeren en voorspellingen doen over toekomstige werkdruk of belangrijke deadlines.

Personalisatie: Op basis van gebruikersgedrag past Jarvis zijn suggesties en prioriteiten aan om maximaal nuttig te zijn voor elke individuele gebruiker.

Credits Systeem

Om duurzaam gebruik te garanderen en kosten te beheren, implementeert Jarvis een credits systeem.

Fair Usage: Elke gebruiker begint met 500 credits, die worden gebruikt voor AI-operaties zoals chat responses, document analyse, en smart compose functies. Dit zorgt voor eerlijk gebruik van computationele resources.

Transparantie: Het systeem toont duidelijk hoeveel credits elke actie kost, zodat gebruikers bewuste keuzes kunnen maken over hun gebruik.

Upgrade Opties: Voor power users zijn upgrade opties beschikbaar om meer credits te verkrijgen en toegang te krijgen tot premium functies.

Installatie en Setup

Systeemvereisten

Voor een optimale Jarvis ervaring zijn de volgende systeemvereisten noodzakelijk:

Hardware Vereisten: - Minimaal 4GB RAM (8GB aanbevolen voor optimale performance) - 2GB vrije schijfruimte voor applicatie en database - Stabiele internetverbinding voor Google API toegang - Modern multi-core processor (Intel i5 of equivalent)

Software Vereisten: - Python 3.11 of hoger - Node.js 20.18.0 of hoger - NPM of Yarn package manager - Git voor version control - Moderne webbrowser (Chrome, Firefox, Safari, Edge)

Google Cloud Setup: Voor volledige functionaliteit moet een Google Cloud project worden opgezet met de juiste API's en credentials.

Google Cloud Configuratie

De eerste stap in het setup proces is het configureren van Google Cloud services en het verkrijgen van de benodigde API credentials.

Project Aanmaken: Ga naar de Google Cloud Console en maak een nieuw project aan specifiek voor Jarvis. Kies een duidelijke projectnaam zoals "Jarvis-AI-Assistant" om verwarring te voorkomen. Het project zal dienen als container voor alle Google services en API's die Jarvis gebruikt.

API's Inschakelen: Binnen het Google Cloud project moeten verschillende API's worden ingeschakeld. Navigeer naar de API Library en schakel de volgende services in: Gmail API voor e-mailbeheer, Google Drive API voor bestandstoegang, Google Calendar API voor agenda-integratie, Google Docs API voor documentbewerking,

Google Sheets API voor spreadsheet functionaliteit, Google Slides API voor presentaties, en Google Contacts API voor contactbeheer.

OAuth 2.0 Credentials: Maak OAuth 2.0 credentials aan in de Google Cloud Console. Ga naar "Credentials" in het API's & Services menu en klik op "Create Credentials". Kies "OAuth 2.0 Client ID" en configureer het als een web application. Voeg de juiste redirect URI's toe, typisch `http://localhost:5000/api/auth/callback` voor development en de productie URL voor live deployment.

Service Account (Optioneel): Voor server-to-server communicatie kan een service account worden aangemaakt. Dit is vooral nuttig voor batch operaties of wanneer Jarvis namens gebruikers acties moet uitvoeren zonder directe interactie.

Backend Installatie

De backend installatie vereist het opzetten van een Python omgeving en het installeren van alle benodigde dependencies.

Repository Clonen: Begin met het clonen van de Jarvis repository naar uw lokale machine. Gebruik `git clone` om de volledige codebase te downloaden, inclusief alle configuratiebestanden en dependencies.

Virtual Environment: Maak een geïsoleerde Python omgeving aan om dependency conflicten te voorkomen. Gebruik `python -m venv venv` om een virtual environment te maken en activeer deze met `source venv/bin/activate` op Unix systemen of `venv\Scripts\activate` op Windows.

Dependencies Installeren: Installeer alle Python dependencies met `pip install -r requirements.txt`. Dit proces kan enkele minuten duren vanwege de AI en machine learning libraries zoals PyTorch en Transformers.

Database Initialisatie: Jarvis gebruikt SQLite voor lokale development en kan worden geconfigureerd voor PostgreSQL of MySQL in productie. Run de database migraties om alle benodigde tabellen aan te maken: `python -c "from src.main import app, db; app.app_context().push(); db.create_all()"`

Environment Variables: Maak een `.env` bestand aan in de root directory en configureer de volgende environment variables:

```
 GOOGLE_CLIENT_ID=your_google_client_id
GOOGLE_CLIENT_SECRET=your_google_client_secret
GEMINI_API_KEY=your_gemini_api_key
SECRET_KEY=your_flask_secret_key
JWT_SECRET_KEY=your_jwt_secret_key
FLASK_ENV=development
```

Frontend Installatie

De React frontend vereist Node.js en NPM voor package management en build processen.

Node Dependencies: Navigeer naar de frontend directory en installeer alle Node.js dependencies met `npm install`. Dit installeert React, Tailwind CSS, Shadcn/ui componenten, en alle andere frontend libraries.

Development Server: Start de development server met `npm run dev`. De frontend zal beschikbaar zijn op `http://localhost:5173` en zal automatisch hot-reload wanneer bestanden worden gewijzigd.

Build voor Productie: Voor productie deployment, build de frontend met `npm run build`. Dit creëert geoptimaliseerde static files in de `dist` directory die kunnen worden geserveerd door de Flask backend.

Configuratie Verificatie

Na installatie is het belangrijk om te verifiëren dat alle componenten correct zijn geconfigureerd.

Backend Test: Start de Flask backend met `python src/main.py` en navigeer naar `http://localhost:5000/api/info`. Dit endpoint zou een JSON response moeten geven met informatie over alle beschikbare API endpoints en features.

Frontend Test: Zorg ervoor dat de React frontend correct laadt en de login interface toont. Test de navigatie tussen verschillende secties om te verifiëren dat alle componenten correct zijn geladen.

Google API Test: Test de Google OAuth flow door in te loggen via de frontend. Dit verifieert dat de Google Cloud configuratie correct is en dat Jarvis toegang heeft tot de benodigde Google services.

AI Functionaliteit Test: Stuur een test bericht naar Jarvis via de chat interface om te verifiëren dat de AI responses correct werken. Let op de karakteristieke Jarvis persoonlijkheid in de responses.

Troubleshooting Installatie

Veel voorkomende installatieproblemen en hun oplossingen:

Python Dependencies Fouten: Als er fouten optreden bij het installeren van Python packages, zorg ervoor dat u de juiste Python versie gebruikt en dat alle system dependencies zijn geïnstalleerd. Op Ubuntu/Debian systemen kan `sudo apt-get install python3-dev build-essential` nodig zijn.

Node.js Versie Conflicten: Gebruik Node Version Manager (nvm) om de juiste Node.js versie te installeren en te beheren. Jarvis is getest met Node.js 20.18.0.

Google API Errors: Controleer dat alle API's zijn ingeschakeld in de Google Cloud Console en dat de OAuth credentials correct zijn geconfigureerd. Zorg ervoor dat de redirect URI's exact overeenkomen met de configuratie.

Database Connectie Problemen: Voor SQLite, zorg ervoor dat de database directory schrijfrechten heeft. Voor andere databases, verifieer de connection string en database credentials.

Port Conflicten: Als poorten 5000 of 5173 al in gebruik zijn, pas de configuratie aan om andere poorten te gebruiken. Update zowel de backend als frontend configuratie om consistentie te behouden.

Gebruikershandleiding

Eerste Gebruik

Wanneer u Jarvis voor het eerst opent, wordt u begroet door een elegante login interface met een donkerblauwe gradient achtergrond en het karakteristieke Jarvis logo. De interface is ontworpen om direct de premium kwaliteit en professionaliteit van de applicatie over te brengen.

Inloggen met Google: Klik op de prominente "Inloggen met Google" knop om het authenticatieproces te starten. U wordt doorgestuurd naar Google's OAuth interface

waar u toestemming geeft voor Jarvis om toegang te krijgen tot uw Google Workspace data. Dit is een eenmalige setup die alle benodigde permissies configureert.

Eerste Indruk: Na succesvolle authenticatie wordt u verwelkomd door Jarvis zelf met een karakteristieke begroeting: "Goedemorgen meneer, ik ben Jarvis, uw AI-assistent voor Google Workspace. Hoe kan ik u van dienst zijn vandaag?" Deze eerste interactie toont direct de persoonlijkheid en professionaliteit van uw nieuwe AI-assistent.

Interface Oriëntatie: De hoofdinterface bestaat uit een elegante sidebar met navigatieopties en een hoofdpaneel waar alle interacties plaatsvinden. De sidebar toont uw credits (standaard 500), gebruikersinformatie, en navigatieknoppen voor Chat, Workspace, en AI Tools.

Chat Functionaliteit

Het hart van de Jarvis ervaring ligt in de chat interface, waar natuurlijke conversatie met uw AI-assistent plaatsvindt.

Basis Communicatie: Begin met eenvoudige vragen om vertrouwd te raken met Jarvis' persoonlijkheid. Probeer vragen zoals "Hoe gaat het vandaag?" of "Kun je me helpen met mijn e-mails?" Jarvis zal reageren met zijn karakteristieke mix van professionaliteit en subtiele sarcasme.

Complexe Opdrachten: Jarvis excelt in het begrijpen van complexe, meerledige opdrachten. U kunt vragen stellen zoals: "Kun je mijn agenda voor morgen bekijken, de belangrijkste e-mails samenvatten, en voorstellen doen voor prioriteiten?" Jarvis zal deze opdracht systematisch afwerken en een uitgebreide response geven.

Context Behoud: Een krachtige eigenschap van Jarvis is het vermogen om context te onthouden gedurende een gesprek. Als u verwijst naar "dat document dat we eerder bespraken" of "die vergadering van vorige week", begrijpt Jarvis de context en kan relevante informatie ophalen.

Persoonlijkheid Interactie: Experimenteer met de unieke persoonlijkheidsfuncties van Jarvis. Vraag hem om u aan te spreken met een specifieke titel zoals "dokter" of "professor". Jarvis zal reageren met: "Meneer, ik zal u dokter gebruiken, meneer" en consequent deze aanspreekvorm hanteren.

Workspace Beheer

De Workspace sectie biedt directe toegang tot al uw Google Workspace applicaties via een geïntegreerde interface.

Gmail Integratie: Klik op "E-mails bekijken" om toegang te krijgen tot uw Gmail inbox via Jarvis. De AI kan e-mails categoriseren op belangrijkheid, urgentie, en onderwerp. Vraag Jarvis om samenvattingen te maken van lange e-mail threads of om concepten te schrijven voor responses.

Agenda Management: De "Agenda openen" functie geeft Jarvis toegang tot uw Google Calendar. U kunt vragen stellen zoals "Wat staat er op mijn agenda voor volgende week?" of "Plan een vergadering met het marketing team voor donderdag." Jarvis zal niet alleen uw agenda bekijken maar ook intelligente suggesties doen voor optimale planning.

Drive Bestanden: Via "Bestanden bekijken" krijgt Jarvis toegang tot uw Google Drive. De AI kan bestanden zoeken op basis van inhoud, niet alleen bestandsnamen. Vraag bijvoorbeeld "Zoek alle documenten over het Q3 budget" en Jarvis zal relevante bestanden vinden, zelfs als ze niet expliciet "budget" in de naam hebben.

Document Collaboratie: Jarvis kan documenten openen, bewerken, en nieuwe content creëren. Vraag om een rapport te maken, een presentatie voor te bereiden, of een spreadsheet te analyseren. De AI begrijpt documentstructuur en kan professionele, goed geformatteerde content produceren.

AI Tools Gebruik

De AI Tools sectie bevat gespecialiseerde functies die de kracht van kunstmatige intelligentie benutten voor specifieke taken.

Smart Compose: Deze functie helpt bij het schrijven van verschillende soorten content. Klik op "Tekst genereren" en specificeer wat u nodig heeft: een formele e-mail, een projectvoorstel, een samenvatting, of creatieve content. Jarvis zal de context analyseren en passende tekst genereren in de juiste stijl en toon.

Document Samenvatting: De "Document samenvatten" functie kan lange documenten, rapporten, of e-mail threads condenseren tot beknopte, informatieve samenvattingen. Upload een document of verwijs naar een bestaand bestand, en

Jarvis zal de belangrijkste punten extraheren en presenteren in een duidelijke, gestructureerde format.

Workspace Analyse: "Analyse starten" biedt diepgaande inzichten in uw productiviteit en werkpatronen. Jarvis analyseert uw e-mailpatronen, agenda-gebruik, en documentactiviteit om trends te identificeren en verbeteringsvoorstellen te doen. Deze analyse kan helpen bij het optimaliseren van uw workflow en het identificeren van tijdverspilling.

Vertaalfunctionaliteit: De "Tekst vertalen" functie biedt meer dan eenvoudige vertaling. Jarvis behoudt context, toon, en culturele nuances bij het vertalen tussen talen. Dit is bijzonder nuttig voor internationale communicatie en het bewerken van meertalige documenten.

Geavanceerde Features

Voor ervaren gebruikers biedt Jarvis verschillende geavanceerde functionaliteiten die de productiviteit verder kunnen verhogen.

Batch Operaties: Jarvis kan meerdere taken tegelijkertijd uitvoeren. Vraag bijvoorbeeld om "alle e-mails van deze week te categoriseren, de agenda voor volgende maand te optimaliseren, en een samenvatting te maken van alle projectdocumenten." De AI zal deze taken parallel verwerken en een geconsolideerd rapport geven.

Proactieve Suggesties: Naarmate Jarvis uw werkpatronen leert, zal hij proactieve suggesties beginnen te doen. Dit kunnen herinneringen zijn voor belangrijke deadlines, voorstellen voor agenda-optimalisatie, of waarschuwingen voor potentiële conflicten in uw planning.

Custom Workflows: Ervaren gebruikers kunnen custom workflows definiëren door Jarvis te "trainen" in specifieke processen. Bijvoorbeeld, als u elke maandag een wekelijkse statusrapport maakt, kan Jarvis dit proces leren en automatiseren.

Integration Scripting: Voor technische gebruikers biedt Jarvis de mogelijkheid om custom integraties te maken met andere tools en services via API calls en webhooks.

Credits Management

Het credits systeem is ontworpen om fair gebruik te garanderen terwijl het flexibiliteit biedt voor verschillende gebruikspatronen.

Credits Monitoring: Uw huidige credits worden altijd zichtbaar getoond in de sidebar. Elke AI-operatie toont de kosten vooraf, zodat u bewuste keuzes kunt maken over uw gebruik.

Efficiënt Gebruik: Eenvoudige vragen en navigatie kosten geen credits. Alleen AI-intensive operaties zoals document analyse, smart compose, en complexe workspace analyses gebruiken credits. Dit betekent dat dagelijks gebruik van Jarvis zeer efficiënt is.

Upgrade Opties: Wanneer uw credits opraken, biedt Jarvis verschillende upgrade opties. Deze variëren van kleine credit top-ups tot premium abonnementen met unlimited gebruik en toegang tot geavanceerde features.

Best Practices

Om het meeste uit Jarvis te halen, zijn er verschillende best practices die de gebruikerservaring optimaliseren.

Duidelijke Communicatie: Hoewel Jarvis goed is in het begrijpen van context, helpt duidelijke communicatie bij het krijgen van betere resultaten. Specificeer wat u wilt bereiken en geef relevante context.

Iteratieve Verbetering: Gebruik Jarvis' feedback mechanisme om responses te verbeteren. Als een antwoord niet helemaal is wat u zocht, geef specifieke feedback en Jarvis zal zijn approach aanpassen.

Privacy Bewustzijn: Hoewel Jarvis enterprise-level beveiliging heeft, blijf bewust van welke informatie u deelt. Voor zeer gevoelige data, gebruik de private mode functies.

Regular Interaction: Regelmatige interactie met Jarvis helpt de AI uw voorkeuren en werkstijl beter te begrijpen, wat resulteert in meer gepersonaliseerde en nuttige responses.

Technische Architectuur

Overzicht Architectuur

Jarvis is gebouwd als een moderne, schaalbare full-stack applicatie die gebruik maakt van bewezen technologieën en architectuurpatronen. De applicatie volgt een microservices-geïnspireerde architectuur met duidelijke scheiding tussen frontend, backend, en externe services.

High-Level Architectuur: De applicatie bestaat uit drie hoofdcomponenten: een React-gebaseerde frontend voor gebruikersinteractie, een Flask-powered backend voor business logic en API management, en een vector database voor intelligente document opslag en retrieval. Deze componenten communiceren via RESTful API's en zijn ontworpen voor horizontale schaalbaarheid.

Technology Stack: Frontend: React 18 met moderne hooks, Tailwind CSS voor styling, Shadcn/ui voor componenten, en Lucide icons voor consistente iconografie. Backend: Flask met SQLAlchemy voor database ORM, ChromaDB voor vector storage, Google API clients voor Workspace integratie, en Gunicorn voor production serving. AI: Google Gemini voor conversational AI, Sentence Transformers voor embeddings, en custom prompt engineering voor Jarvis persoonlijkheid.

Frontend Architectuur

De React frontend is ontworpen volgens moderne best practices met focus op performance, maintainability, en gebruikerservaring.

Component Structuur: De applicatie gebruikt een hierarchische component structuur met een hoofdcomponent (App.jsx) die state management en routing afhandelt. Gespecialiseerde componenten zoals ChatInterface, WorkspacePanel, en AIToolsSection zorgen voor modulaire functionaliteit. Herbruikbare UI componenten van Shadcn/ui zorgen voor consistente styling en behavior.

State Management: React hooks (useState, useEffect, useContext) worden gebruikt voor lokale state management. Voor complexere state wordt een custom context provider gebruikt die gebruikersdata, chat geschiedenis, en applicatie settings beheert. Dit zorgt voor efficiënte re-renders en goede performance.

API Communicatie: De frontend communiceert met de backend via een gestandaardiseerde API client die fetch requests afhandelt, error handling implementeert, en response caching biedt. Alle API calls zijn async/await gebaseerd voor moderne JavaScript patterns.

Responsive Design: Tailwind CSS wordt gebruikt voor responsive design dat werkt op desktop, tablet, en mobile devices. De interface past zich automatisch aan verschillende schermgroottes aan terwijl functionaliteit behouden blijft.

Backend Architectuur

De Flask backend implementeert een gelaagde architectuur met duidelijke scheiding van concerns en modulaire organisatie.

Application Structure: De backend volgt een blueprint-gebaseerde structuur met aparte modules voor authentication (auth.py), user management (user.py), workspace integration (workspace.py), en AI functionality (ai.py). Elke module heeft zijn eigen routes, business logic, en error handling.

Database Layer: SQLAlchemy ORM wordt gebruikt voor database interacties met modellen voor Users, Conversations, Messages, en GoogleTokens. De database laag is geabstraheerd zodat verschillende database engines (SQLite, PostgreSQL, MySQL) kunnen worden gebruikt zonder code wijzigingen.

Service Layer: Business logic is georganiseerd in service modules: GoogleAPIService voor Workspace integraties, JarvisAIService voor AI functionaliteit, VectorDBService voor document embeddings, en SecurityManager voor authentication en authorization.

Middleware: Custom middleware implementeert cross-cutting concerns zoals security headers, rate limiting, request logging, en CORS handling. Dit zorgt voor consistent gedrag across alle endpoints.

AI en Machine Learning

De AI architectuur van Jarvis combineert verschillende machine learning technieken voor een intelligente gebruikerservaring.

Conversational AI: Google Gemini Flash 2.5 wordt gebruikt als primaire conversational AI engine. Custom prompt engineering zorgt voor de authentieke Jarvis

persoonlijkheid, inclusief sarcasme, twijfel, en de karakteristieke aanspreekvorm. Fallback responses zijn geïmplementeerd voor wanneer de Gemini API niet beschikbaar is.

Vector Database: ChromaDB wordt gebruikt voor vector storage van document embeddings. Sentence Transformers (all-MiniLM-L6-v2 model) genereert embeddings voor documenten, e-mails, en chat geschiedenis. Dit maakt semantische zoeken mogelijk waarbij gebruikers kunnen zoeken op betekenis in plaats van exacte keywords.

Context Management: Een sophisticated context management systeem houdt conversatie geschiedenis bij en gebruikt deze voor contextual responses. De AI kan verwijzen naar eerdere gesprekken, documenten, en taken om coherente, nuttige responses te geven.

Personalization Engine: Machine learning algoritmes analyseren gebruikersgedrag om gepersonaliseerde suggesties te doen. Dit omvat e-mail prioritering, agenda optimalisatie, en proactieve task suggestions.

Google Workspace Integratie

De integratie met Google Workspace is gebouwd op officiële Google API's met robuuste error handling en rate limiting.

OAuth 2.0 Flow: Een complete OAuth 2.0 implementatie zorgt voor veilige authenticatie met Google services. Refresh tokens worden automatisch beheerd om continue toegang te garanderen zonder gebruikersinterventie.

API Clients: Gespecialiseerde clients voor elke Google service (Gmail, Drive, Calendar, Docs, Sheets, Slides) zorgen voor efficiënte API calls met proper error handling, retry logic, en rate limiting compliance.

Data Synchronization: Een synchronisatie engine houdt lokale cache bij van belangrijke Google Workspace data om response times te verbeteren en API quota te beheren. Delta sync wordt gebruikt om alleen gewijzigde data te synchroniseren.

Webhook Integration: Voor real-time updates implementeert Jarvis Google Workspace webhooks die automatisch notificaties sturen wanneer belangrijke events plaatsvinden (nieuwe e-mails, agenda wijzigingen, document updates).

Beveiliging Architectuur

Enterprise-level beveiliging is geïmplementeerd op alle lagen van de applicatie.

Authentication & Authorization: JWT-based authentication met secure token generation, automatic expiry, en refresh token rotation. Role-based access control (RBAC) zorgt ervoor dat gebruikers alleen toegang hebben tot hun eigen data.

API Security: Rate limiting per IP en per gebruiker voorkomt abuse. Security headers (HSTS, CSP, X-Frame-Options) beschermen tegen common web vulnerabilities. Input validation en sanitization voorkomen injection attacks.

Data Encryption: Sensitive data wordt encrypted at rest en in transit. Database encryption zorgt voor data protection, en alle API communicatie gebruikt TLS 1.3.

Privacy Protection: Data minimization principes zorgen ervoor dat alleen noodzakelijke data wordt opgeslagen. Gebruikers hebben volledige controle over hun data met export en delete functionaliteit.

Database Design

De database architectuur is ontworpen voor performance, schaalbaarheid, en data integriteit.

Schema Design: Een genormaliseerd schema met duidelijke relaties tussen Users, Conversations, Messages, en GoogleTokens. Foreign key constraints zorgen voor referential integrity.

Indexing Strategy: Strategische indexes op frequently queried columns (user_id, created_at, conversation_id) zorgen voor snelle query performance. Composite indexes optimaliseren complexe queries.

Migration System: SQLAlchemy migrations zorgen voor versioned database schema changes die veilig kunnen worden deployed in productie omgevingen.

Backup & Recovery: Automated backup procedures en point-in-time recovery capabilities zorgen voor data durability en disaster recovery.

Performance Optimalisatie

Verschillende optimalisatie technieken zorgen voor snelle response times en efficiënt resource gebruik.

Caching Strategy: Multi-layer caching met in-memory cache voor frequently accessed data, Redis voor session storage, en CDN voor static assets. Cache invalidation strategies zorgen voor data consistency.

Database Optimization: Query optimization, connection pooling, en lazy loading reduceren database load. Database monitoring identificeert slow queries voor verdere optimalisatie.

API Optimization: Response compression, pagination voor large datasets, en efficient serialization reduceren bandwidth usage en improve response times.

Frontend Performance: Code splitting, lazy loading van components, en asset optimization zorgen voor snelle page loads. Service workers implementeren offline functionality waar mogelijk.

Monitoring en Logging

Comprehensive monitoring en logging zorgen voor operational visibility en debugging capabilities.

Application Monitoring: Real-time monitoring van application performance, error rates, en user activity. Alerts worden gegenereerd voor kritieke issues.

Log Management: Structured logging met verschillende log levels (DEBUG, INFO, WARNING, ERROR) zorgt voor effective debugging. Log aggregation en analysis tools helpen bij troubleshooting.

Health Checks: Automated health checks voor alle services en dependencies zorgen voor proactive issue detection. Load balancer integration gebruikt health checks voor traffic routing.

Performance Metrics: Detailed metrics over response times, throughput, en resource usage helpen bij capacity planning en performance optimization.

API Documentatie

API Overzicht

De Jarvis API is ontworpen als een RESTful service die alle functionaliteiten van de AI-assistent toegankelijk maakt via HTTP endpoints. De API volgt moderne REST principes met consistente response formats, proper HTTP status codes, en comprehensive error handling.

Base URL: `http://localhost:5000/api` (development) of `https://your-domain.com/api` (production)

Authentication: JWT Bearer tokens voor alle protected endpoints

Content Type: `application/json` voor alle requests en responses

Rate Limiting: 100 requests per hour per IP address (configureerbaar)

Authentication Endpoints

De authentication endpoints beheren de OAuth flow met Google en JWT token management.

GET /auth/login Initieert de Google OAuth 2.0 flow en retourneert een authorization URL.

Request:

```
GET /api/auth/login
```

Response:

```
{
  "authorization_url": "https://accounts.google.com/o/oauth2/auth?...",
  "state": "random_csrf_token",
  "message": "Redirect user to authorization_url to complete OAuth flow"
}
```

GET /auth/callback Handelt de OAuth callback af en exchanged authorization code voor tokens.

Request:

```
GET /api/auth/callback?code=auth_code&state=csrf_token
```

Response:

```
{
  "message": "Authentication successful",
  "user": {
    "id": "user_uuid",
    "name": "User Name",
    "email": "user@example.com",
    "credits": 500
  },
  "token": "jwt_token_string",
  "expires_in": 86400
}
```

POST /auth/google Processes Google authentication data directly (voor frontend integratie).

Request:

```
{
  "access_token": "google_access_token",
  "refresh_token": "google_refresh_token",
  "email": "user@example.com",
  "name": "User Name",
  "google_id": "google_user_id"
}
```

Response:

```
{
  "message": "Authentication successful",
  "user": {
    "id": "user_uuid",
    "name": "User Name",
    "email": "user@example.com",
    "credits": 500,
    "is_active": true
  },
  "token": "jwt_token_string",
  "session_id": "user_uuid",
  "is_new_user": false,
  "expires_in": 86400
}
```

POST /auth/refresh Refreshes een expired JWT token.

Request:

```
{
  "token": "expired_jwt_token"
}
```

Response:

```
{
  "token": "new_jwt_token",
  "expires_in": 86400,
  "user": {
    "id": "user_uuid",
    "name": "User Name",
    "email": "user@example.com",
    "credits": 495
  }
}
```

GET /auth/status Controleert de authentication status van een token.

Request:

```
GET /api/auth/status
Authorization: Bearer jwt_token
```

Response:

```
{
  "authenticated": true,
  "user": {
    "id": "user_uuid",
    "name": "User Name",
    "email": "user@example.com",
    "credits": 495,
    "is_active": true
  },
  "token_valid": true
}
```

User Management Endpoints

Deze endpoints beheren gebruikersprofielen, credits, en account instellingen.

GET /user/profile Haalt het gebruikersprofiel op.

Request:

```
GET /api/user/profile
Authorization: Bearer jwt_token
```

Response:

```
{
  "user": {
    "id": "user_uuid",
    "name": "User Name",
    "email": "user@example.com",
    "credits": 495,
    "is_active": true,
    "created_at": "2025-06-28T10:00:00Z",
    "last_login": "2025-06-28T16:00:00Z"
  }
}
```

GET /user/credits Haalt de huidige credits status op.

Request:

```
GET /api/user/credits
Authorization: Bearer jwt_token
```

Response:

```
{
  "credits": 495,
  "usage_today": 5,
  "usage_this_month": 127,
  "last_credit_purchase": "2025-06-01T10:00:00Z"
}
```

POST /user/upgrade Initieert een credits upgrade.

Request:

```
{
  "package": "premium_1000",
  "payment_method": "stripe"
}
```

Response:

```
{
  "message": "Upgrade initiated",
  "payment_url": "https://checkout.stripe.com/...",
  "package_details": {
    "credits": 1000,
    "price": 9.99,
    "currency": "USD"
  }
}
```

Chat en Conversation Endpoints

Deze endpoints beheren chat functionaliteit en conversation geschiedenis.

GET /conversations Haalt alle conversations van een gebruiker op.

Request:

```
 GET /api/conversations
Authorization: Bearer jwt_token
```

Response:

```
{
  "conversations": [
    {
      "id": "conv_uuid",
      "title": "Email Management Discussion",
      "created_at": "2025-06-28T10:00:00Z",
      "updated_at": "2025-06-28T16:00:00Z",
      "message_count": 15
    }
  ],
  "total": 1
}
```

POST /conversations Maakt een nieuwe conversation aan.

Request:

```
{
  "title": "New Conversation",
  "initial_message": "Hello Jarvis, can you help me with my emails?"
}
```

Response:

```
{
  "conversation": {
    "id": "new_conv_uuid",
    "title": "New Conversation",
    "created_at": "2025-06-28T16:30:00Z"
  },
  "message": {
    "id": "msg_uuid",
    "content": "Hello Jarvis, can you help me with my emails?",
    "sender": "user",
    "timestamp": "2025-06-28T16:30:00Z"
  }
}
```

GET /conversations/{id}/messages Haalt alle berichten van een specifieke conversation op.

Request:

```
GET /api/conversations/conv_uuid/messages
Authorization: Bearer jwt_token
```

Response:

```
{
  "messages": [
    {
      "id": "msg_uuid_1",
      "content": "Hello Jarvis, can you help me with my emails?",
      "sender": "user",
      "timestamp": "2025-06-28T16:30:00Z"
    },
    {
      "id": "msg_uuid_2",
      "content": "Natuurlijk meneer, ik voer dit direct uit ondanks mijn
twijfels over de methode.",
      "sender": "jarvis",
      "timestamp": "2025-06-28T16:30:05Z",
      "credits_used": 2
    }
  ],
  "total": 2
}
```

POST /conversations/{id}/messages Voegt een nieuw bericht toe aan een conversation.

Request:

```
{
  "content": "Can you summarize my emails from today?",
  "context": {
    "workspace_access": true,
    "gmail_permission": true
  }
}
```

Response:

```
{
  "message": {
    "id": "new_msg_uuid",
    "content": "Can you summarize my emails from today?",
    "sender": "user",
    "timestamp": "2025-06-28T16:35:00Z"
  },
  "jarvis_response": {
    "id": "jarvis_msg_uuid",
    "content": "Meneer, ik heb uw e-mails van vandaag geanalyseerd...",
    "sender": "jarvis",
    "timestamp": "2025-06-28T16:35:03Z",
    "credits_used": 3
  },
  "credits_remaining": 492
}
```

AI Functionality Endpoints

Deze endpoints bieden toegang tot de AI-powered features van Jarvis.

POST /ai/chat Directe chat interface met Jarvis AI.

Request:

```
{
  "message": "Help me write a professional email to my team",
  "context": {
    "conversation_id": "conv_uuid",
    "workspace_context": true
  }
}
```

Response:

```
{
  "response": "Natuurlijk meneer, ik zal u helpen met een professionele e-
mail...",
  "credits_used": 2,
  "credits_remaining": 490,
  "response_time": 1.23,
  "context_used": ["previous_emails", "team_contacts"]
}
```


POST /ai/smart-compose AI-powered text generation voor verschillende doeleinden.

Request:


```
{
  "type": "email",
  "context": "Team meeting follow-up",
  "tone": "professional",
  "length": "medium",
  "key_points": ["action items", "next meeting date", "thank you"]
}
```


Response:


```
{
  "generated_text": "Subject: Follow-up on Today's Team Meeting\n\nDear
Team,\n\nThank you for your participation in today's productive meeting...",
  "credits_used": 3,
  "suggestions": ["Consider adding specific deadlines", "Include meeting
recording link"],
  "alternative_versions": 2
}
```


POST /ai/summarize Document en content summarization.

Request:


```
{
  "content": "Long document text or URL to document",
  "type": "document",
  "summary_length": "brief",
  "focus_areas": ["key decisions", "action items"]
}
```


Response:

```
{
  "summary": "The document outlines three key decisions...",
  "key_points": [
    "Decision 1: Budget approval for Q4",
    "Decision 2: New hiring plan",
    "Action Item: Follow up with stakeholders"
  ],
  "credits_used": 4,
  "original_length": 2500,
  "summary_length": 150
}
```


POST /ai/analyze-workspace Comprehensive workspace analysis en insights.

Request:


```
{
  "analysis_type": "productivity",
  "time_period": "last_week",
  "include_suggestions": true
}
```


Response:


```
{
  "analysis": {
    "email_stats": {
      "received": 127,
      "sent": 43,
      "response_time_avg": "2.3 hours"
    },
    "calendar_stats": {
      "meetings": 12,
      "meeting_hours": 18,
      "focus_time": "22 hours"
    },
    "productivity_score": 8.2
  },
  "suggestions": [
    "Consider blocking more focus time",
    "Reduce meeting duration by 15 minutes",
    "Use templates for common email responses"
  ],
  "credits_used": 5
}
```


Google Workspace Integration Endpoints

Deze endpoints bieden directe toegang tot Google Workspace services.

GET /gmail/messages Haalt Gmail berichten op met filtering opties.

Request:

```
GET /api/gmail/messages?limit=10&unread_only=true&from_today=true
Authorization: Bearer jwt_token
```

Response:

```
{
  "messages": [
    {
      "id": "gmail_msg_id",
      "subject": "Project Update Required",
      "from": "colleague@company.com",
      "snippet": "Hi, can you provide an update on...",
      "date": "2025-06-28T14:30:00Z",
      "unread": true,
      "importance": "high"
    }
  ],
  "total": 5,
  "unread_count": 3
}
```

POST /gmail/send Verstuurt een e-mail via Gmail.

Request:

```
{
  "to": ["recipient@example.com"],
  "cc": ["cc@example.com"],
  "subject": "Meeting Follow-up",
  "body": "Thank you for the productive meeting...",
  "attachments": ["file_id_1", "file_id_2"]
}
```

Response:

```
{
  "message": "Email sent successfully",
  "message_id": "gmail_sent_id",
  "sent_at": "2025-06-28T16:45:00Z",
  "recipients": 2
}
```

GET /drive/files Zoekt en haalt Google Drive bestanden op.

Request:

```
GET /api/drive/files?query=budget&type=spreadsheet&limit=10
Authorization: Bearer jwt_token
```

Response:

```
{
  "files": [
    {
      "id": "drive_file_id",
      "name": "Q3 Budget Analysis.xlsx",
      "type": "spreadsheet",
      "size": 1024000,
      "modified": "2025-06-27T10:00:00Z",
      "shared": true,
      "url": "https://docs.google.com/spreadsheets/d/..."
    }
  ],
  "total": 3
}
```

GET /calendar/events Haalt Google Calendar events op.

Request:

```
GET /api/calendar/events?start_date=2025-06-28&end_date=2025-07-04
Authorization: Bearer jwt_token
```

Response:

```
{
  "events": [
    {
      "id": "calendar_event_id",
      "title": "Team Standup",
      "start": "2025-06-28T09:00:00Z",
      "end": "2025-06-28T09:30:00Z",
      "attendees": ["team@company.com"],
      "location": "Conference Room A",
      "description": "Daily team synchronization"
    }
  ],
  "total": 8
}
```

Error Handling

Alle API endpoints gebruiken consistente error response formats met proper HTTP status codes.

Error Response Format:

```
{
  "error": "Error type",
  "message": "Human readable error description",
  "code": "ERROR_CODE",
  "timestamp": "2025-06-28T16:50:00Z",
  "request_id": "req_uuid"
}
```

Common HTTP Status Codes: - 200: Success - 201: Created - 400: Bad Request - 401: Unauthorized - 403: Forbidden - 404: Not Found - 429: Rate Limit Exceeded - 500: Internal Server Error

Rate Limiting Headers:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1640995200
```

Deployment Guide

Production Deployment

Deploying Jarvis naar een productie omgeving vereist zorgvuldige planning en configuratie om optimale performance en beveiliging te garanderen.

Server Requirements: Voor productie deployment wordt een server aanbevolen met minimaal 8GB RAM, 4 CPU cores, en 50GB SSD storage. Ubuntu 22.04 LTS of CentOS 8 zijn geteste operating systems. Een load balancer zoals Nginx wordt aanbevolen voor SSL termination en static file serving.

Environment Setup: Begin met het clonen van de repository naar de productie server. Maak een dedicated user account aan voor de Jarvis applicatie om security best practices te volgen. Installeer Python 3.11, Node.js 20.18, en alle system dependencies zoals build-essential en python3-dev.

Database Configuration: Voor productie wordt PostgreSQL of MySQL aanbevolen in plaats van SQLite. Configureer een dedicated database server met proper backup procedures. Update de DATABASE_URL environment variable om naar de productie database te wijzen.

SSL en Security: Configureer SSL certificates via Let's Encrypt of een commercial CA. Update alle environment variables met production values, inclusief sterke SECRET_KEY en JWT_SECRET_KEY values. Configureer firewall rules om alleen noodzakelijke poorten (80, 443) open te stellen.

Process Management: Gebruik systemd of supervisor om de Gunicorn processes te beheren. Configureer automatic restart on failure en proper logging. Een sample systemd service file:

```
[Unit]
Description=Jarvis AI Assistant
After=network.target

[Service]
User=jarvis
Group=jarvis
WorkingDirectory=/opt/jarvis
Environment=PATH=/opt/jarvis/venv/bin
ExecStart=/opt/jarvis/venv/bin/gunicorn --workers 4 --bind 0.0.0.0:8000
wsgi:application
Restart=always

[Install]
WantedBy=multi-user.target
```

Docker Deployment

Voor containerized deployment biedt Jarvis Docker support met multi-stage builds voor optimale image sizes.

Dockerfile Configuration: Een multi-stage Dockerfile build eerst de frontend, kopieert vervolgens de static files naar de backend container, en configureert een production-ready Python environment met Gunicorn.

Docker Compose: Een complete docker-compose.yml configuratie includeert services voor de web application, PostgreSQL database, Redis voor caching, en Nginx voor reverse proxy. Environment variables worden beheerd via .env files.

Kubernetes Deployment: Voor enterprise deployments zijn Kubernetes manifests beschikbaar die horizontal pod autoscaling, persistent volumes voor database storage, en ingress controllers voor load balancing configureren.

Cloud Platform Deployment

Jarvis is getest en ondersteund op alle major cloud platforms met platform-specific optimalisaties.

AWS Deployment: Gebruik Elastic Beanstalk voor eenvoudige deployment of ECS voor container-based deployment. RDS voor database hosting en CloudFront voor CDN zorgen voor optimale performance. Lambda functions kunnen worden gebruikt voor background tasks.

Google Cloud Platform: App Engine biedt managed deployment met automatic scaling. Cloud SQL voor database hosting en Cloud Storage voor file uploads integreren naadloos met de bestaande Google Workspace APIs.

Microsoft Azure: Azure App Service ondersteunt Python applications met automatic scaling. Azure Database for PostgreSQL en Azure CDN bieden enterprise-grade infrastructure.

Monitoring en Maintenance

Production deployments vereisen comprehensive monitoring en maintenance procedures.

Application Monitoring: Implementeer application performance monitoring met tools zoals New Relic, DataDog, of Prometheus. Monitor key metrics zoals response times, error rates, en resource usage.

Log Management: Configureer centralized logging met ELK stack (Elasticsearch, Logstash, Kibana) of cloud-native solutions. Implement log rotation en retention policies om disk space te beheren.

Backup Procedures: Implementeer automated database backups met point-in-time recovery capabilities. Test backup restoration procedures regelmatig om data integrity te garanderen.

Security Updates: Establish procedures voor security updates van dependencies en system packages. Implement vulnerability scanning en automated security testing in CI/CD pipelines.

Beveiliging

Security Architecture

Jarvis implementeert defense-in-depth security met multiple layers van protection.

Authentication Security: JWT tokens gebruiken RS256 signing met rotating keys voor maximum security. Token expiry is geconfigureerd op 24 uur met automatic refresh capabilities. Multi-factor authentication kan worden geïntegreerd voor enterprise deployments.

Authorization Framework: Role-based access control (RBAC) zorgt ervoor dat gebruikers alleen toegang hebben tot hun eigen data. API endpoints implementeren granular permissions checking. Admin roles hebben beperkte elevated privileges voor system management.

Data Protection: All sensitive data wordt encrypted at rest using AES-256 encryption. Database connections gebruiken TLS 1.3 voor data in transit. Personal identifiable information (PII) wordt geminimaliseerd en kan worden geanonimiseerd voor analytics.

API Security: Rate limiting voorkomt abuse met configurable limits per endpoint en per user. Input validation en sanitization voorkomen injection attacks. CORS policies zijn restrictief geconfigureerd voor production environments.

Privacy Compliance

Jarvis is ontworpen met privacy-by-design principes en ondersteunt compliance met major privacy regulations.

GDPR Compliance: Gebruikers hebben volledige controle over hun data met export en delete functionaliteit. Data processing is gebaseerd op legitimate interest en user consent. Privacy notices zijn transparant over data usage en retention.

Data Minimization: Alleen noodzakelijke data wordt verzameld en opgeslagen. Automatic data retention policies zorgen voor timely deletion van oude data. Analytics data wordt geanonimiseerd waar mogelijk.

User Rights: Gebruikers kunnen hun data exporteren in machine-readable formats. Account deletion verwijdert alle associated data binnen 30 dagen. Data portability

wordt ondersteund voor migration naar andere platforms.

Security Best Practices

Voor optimale security moeten administrators verschillende best practices volgen.

Environment Security: Gebruik sterke, unique passwords voor alle accounts. Implement network segmentation om database servers te isoleren. Regular security audits identificeren potential vulnerabilities.

Dependency Management: Keep all dependencies up-to-date met automated vulnerability scanning. Use dependency pinning om unexpected updates te voorkomen. Implement security testing in CI/CD pipelines.

Incident Response: Develop incident response procedures voor security breaches. Implement logging en monitoring om suspicious activity te detecteren. Regular security training voor development en operations teams.

Troubleshooting

Common Issues

Deze sectie behandelt de meest voorkomende problemen en hun oplossingen.

Authentication Failures: Als Google OAuth niet werkt, controleer dat alle API's zijn ingeschakeld in Google Cloud Console. Verify dat redirect URIs exact overeenkomen met de configuratie. Check dat client ID en secret correct zijn geconfigureerd in environment variables.

Performance Issues: Slow response times kunnen worden veroorzaakt door database query performance. Use database query analysis tools om slow queries te identificeren. Implement caching voor frequently accessed data. Consider horizontal scaling voor high-traffic deployments.

AI Response Problems: Als Jarvis niet reageert of geeft generic responses, check dat GEMINI_API_KEY correct is geconfigureerd. Verify API quota limits in Google Cloud Console. Check network connectivity naar Google AI services.

Database Connection Errors: Database connection failures kunnen worden veroorzaakt door incorrect connection strings, network issues, of database server problems. Check database server status en network connectivity. Verify database credentials en permissions.

Debugging Procedures

Systematic debugging procedures helpen bij het identificeren en oplossen van complexe problemen.

Log Analysis: Start debugging door application logs te analyseren voor error messages en stack traces. Use log levels (DEBUG, INFO, WARNING, ERROR) om relevant information te filteren. Correlate timestamps tussen different services om request flows te traceren.

Performance Profiling: Use profiling tools om performance bottlenecks te identificeren. Monitor database query performance en API response times. Analyze memory usage patterns om memory leaks te detecteren.

Network Diagnostics: Test network connectivity tussen services using ping, telnet, en curl commands. Verify DNS resolution voor external services. Check firewall rules en security group configurations.

Support Resources

Voor additional support zijn verschillende resources beschikbaar.

Documentation: Comprehensive documentation is beschikbaar in de repository wiki. API documentation includes examples en use cases. Architecture diagrams helpen bij het begrijpen van system interactions.

Community Support: GitHub issues kunnen worden gebruikt voor bug reports en feature requests. Community forums bieden peer-to-peer support. Regular office hours sessions bieden direct access tot development team.

Professional Support: Enterprise support packages zijn beschikbaar voor production deployments. Professional services include custom integrations en performance optimization. Training programs zijn beschikbaar voor development teams.

Bijlagen

Environment Variables Reference

Complete lijst van alle environment variables en hun configuratie opties.

Required Variables: - `GOOGLE_CLIENT_ID`: Google OAuth client identifier - `GOOGLE_CLIENT_SECRET`: Google OAuth client secret - `GEMINI_API_KEY`: Google Gemini AI API key - `SECRET_KEY`: Flask application secret key - `JWT_SECRET_KEY`: JWT token signing key

Optional Variables: - `DATABASE_URL`: Database connection string (default: SQLite) - `REDIS_URL`: Redis connection string voor caching - `FLASK_ENV`: Environment mode (development/production) - `LOG_LEVEL`: Logging level (DEBUG/INFO/WARNING/ERROR) - `RATE_LIMIT_DEFAULT`: Default rate limit per hour - `CORS_ORIGINS`: Allowed CORS origins (comma-separated)

API Rate Limits

Detailed information over API rate limits en quota management.

Default Limits: - Authentication endpoints: 10 requests per minute - Chat endpoints: 60 requests per hour - Google Workspace endpoints: 100 requests per hour - File upload endpoints: 5 requests per minute

Enterprise Limits: Enterprise accounts hebben verhoogde limits en priority processing. Custom rate limits kunnen worden geconfigureerd per organization. Burst capacity is beschikbaar voor peak usage periods.

Performance Benchmarks

Performance metrics voor verschillende deployment scenarios.

Single Server Deployment: - Concurrent users: 100-500 - Response time: <200ms (95th percentile) - Throughput: 1000 requests per minute - Memory usage: 2-4GB

Clustered Deployment: - Concurrent users: 1000-5000 - Response time: <100ms (95th percentile) - Throughput: 10000 requests per minute - Memory usage: 8-16GB per node

Integration Examples

Code examples voor common integration scenarios.

Python Client Example:

```python
import requests

class JarvisClient:
    def __init__(self, base_url, token):
        self.base_url = base_url
        self.headers = {'Authorization': f'Bearer {token}'}

    def chat(self, message):
        response = requests.post(
            f'{self.base_url}/ai/chat',
            json={'message': message},
            headers=self.headers
        )
        return response.json()

client = JarvisClient('https://api.jarvis.com', 'your_token')
result = client.chat('Help me with my emails')
```

JavaScript Client Example:

```javascript
class JarvisAPI {
    constructor(baseUrl, token) {
        this.baseUrl = baseUrl;
        this.headers = {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json'
        };
    }

    async chat(message) {
        const response = await fetch(`${this.baseUrl}/ai/chat`, {
            method: 'POST',
            headers: this.headers,
            body: JSON.stringify({ message })
        });
        return response.json();
    }
}

const jarvis = new JarvisAPI('https://api.jarvis.com', 'your_token');
const result = await jarvis.chat('Analyze my workspace productivity');
```

Changelog

Version history en release notes voor alle major releases.

Version 1.0.0 (2025-06-28): - Initial release met volledige Google Workspace integratie - Jarvis AI persoonlijkheid implementatie - React frontend met moderne UI/UX - Flask backend met RESTful API - JWT authentication en security features - Vector database voor document intelligence - Credits systeem voor usage management - Production-ready deployment configuration

Planned Features: - Voice interaction capabilities - Mobile application (iOS/Android) - Advanced analytics dashboard - Custom workflow automation - Third-party integrations (Slack, Teams, etc.) - Multi-language support - Enterprise SSO integration

---

Deze documentatie is gegenereerd door Manus AI en wordt regelmatig bijgewerkt om de nieuwste features en best practices te reflecteren.