REPLIT AI AGENT PROMPT - LEX CAO EXPERT SAAS

KOPIEER DEZE VOLLEDIGE TEKST EN PLAK IN REPLIT AI AGENT

Build a complete **Multi-Tenant SaaS platform** called "LEX CAO Expert" for Dutch temp agencies (uitzendbureaus).

CORE PRODUCT

LEX = AI agent that answers CAO (collective labor agreement) questions for payroll workers at temp agencies.

Tech Stack: - Backend: Flask (Python) - Database: PostgreSQL (Replit native) - Frontend: HTML + Vanilla JavaScript + Tailwind CSS - AI: Google Vertex AI (existing agent already deployed) - Payments: Stripe - Email: SendGrid

CRITICAL CONTEXT

1. **LEX agent already exists and works** in Google Vertex AI
2. **70 CAO documents** already in Google Cloud Storage bucket: `uzb-agent-cao-corpus`
3. **RAG (retrieval) happens automatically** in Vertex AI
4. We only need to integrate via API call

BUSINESS MODEL

Multi-Tenant Hierarchy:

```
SUPER ADMIN (me)
  └─ TENANTS (temp agencies)
      └─ TENANT ADMIN
          └─ END USERS (payroll employees)
```

Pricing: - Professional: €499/month (5 users, unlimited questions) - Enterprise: €1.199/month (unlimited users, unlimited questions) - 14-day free trial (no credit card)

Subdomain routing: Each tenant gets `companyname.lex-cao.replit.app`

ENVIRONMENT VARIABLES NEEDED

Create these in Replit Secrets:

```
# Google Cloud (existing setup)
GOOGLE_CLOUD_PROJECT=uzb-agent
VERTEX_AI_LOCATION=europe-west1
VERTEX_AI_AGENT_ID=<your-agent-id>
GOOGLE_APPLICATION_CREDENTIALS=<service-account-json>

# Flask
SECRET_KEY=<random-string>

# Stripe
STRIPE_PUBLIC_KEY=pk_test_...
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...

# SendGrid
SENDGRID_API_KEY=SG...
FROM_EMAIL=noreply@lex-cao.nl

# App
APP_URL=https://lex-cao-expert.replit.app
```

DATABASE SCHEMA

```
-- Super Admin
super_admins (id, email, password_hash, name, created_at)

-- Tenants (temp agencies)
tenants (
    id, company_name, subdomain,
    contact_email, contact_name,
    status (trial/active/suspended),
    trial_ends_at, max_users,
    created_at, updated_at
)

-- Users (payroll employees)
users (
    id, tenant_id, email, password_hash,
    first_name, last_name,
    role (admin/user),
    is_active, created_at
)
```

```
UNIQUE(tenant_id, email) -- email unique per tenant

-- Chats
chats (
    id, tenant_id, user_id, title,
    created_at, updated_at
)

-- Messages
messages (
    id, tenant_id, chat_id,
    role (user/assistant), content,
    created_at
)

-- Subscriptions
subscriptions (
    id, tenant_id,
    plan (professional/enterprise),
    status (trialing/active/past_due/canceled),
    stripe_customer_id, stripe_subscription_id,
    current_period_start, current_period_end,
    created_at
)
```

TENANT ISOLATION (CRITICAL!)

Every query MUST include `tenant_id` to prevent data leaks:

```
# WRONG - data leak!
chats = Chat.query.filter_by(user_id=user_id).all()

# CORRECT - tenant isolated
chats = Chat.query.filter_by(
    tenant_id=g.tenant.id,
    user_id=user_id
).all()
```

Use middleware to set `g.tenant` from: 1. Subdomain (e.g., `companyxyz.lex-cao.replit.app`) 2. Current logged-in user's tenant_id

LEX VERTEX AI INTEGRATION

```
import vertexai
from vertexai.preview import reasoning_engines
import os

vertexai.init(
    project=os.getenv('GOOGLE_CLOUD_PROJECT'),
    location=os.getenv('VERTEX_AI_LOCATION')
)

agent = reasoning_engines.ReasoningEngine(
    agent_id=os.getenv('VERTEX_AI_AGENT_ID')
)

def chat_with_lex(user_message):
    response = agent.query(input=user_message)
    return response.text
```

Important: Vertex AI agent automatically searches the 70 CAO documents via RAG. No manual RAG implementation needed!

LEX LOADING ANIMATION

While waiting for LEX response (20-30 seconds), show animated ASCII character doing funny things.

5 Random Scenarios:

1. **Library:** LEX walks → picks up CAO book → reads → stumbles → gets up
2. **Coffee:** LEX drinks coffee → too hot! → blows on it → continues
3. **Papers:** LEX holds papers → papers fly away → catches them
4. **Calculator:** LEX calculates → wrong button → tries again
5. **Phone:** Phone rings → answers → hangs up → back to work

Implementation:

```
        const LEX_SCENES = [
            {
                frames: [
                    { ascii: "  📚\n /|\\\n / \\", text: "Searching 70 CAO documents..." },
                    { ascii: "   📚\n  /|\\\n  / \\", text: "Hmm, which article was it..." },
                    { ascii: "  📖✨\n  /|\\\n  / \\", text: "Found in Glastuinbouw CAO!" }
                ]
            },
            // ... 4 more scenes
        ];

        function startLexAnimation() {
            const scene = LEX_SCENES[Math.floor(Math.random() * LEX_SCENES.length)];
            // Loop through frames every 3 seconds
        }
```

STRIPE INTEGRATION

Products to create in Stripe Dashboard: - Professional Plan: €499/month (price_id: `price_professional`) - Enterprise Plan: €1.199/month (price_id: `price_enterprise`)

Checkout Flow: 1. User clicks “Upgrade” → backend creates Stripe Checkout Session 2. Redirect to Stripe hosted checkout page 3. After payment → Stripe webhook: `checkout.session.completed` 4. Backend updates: `subscription.status = 'active'`, `tenant.status = 'active'`

Webhook Handler:

```
        @app.route('/webhook/stripe', methods=['POST'])
        def stripe_webhook():
            payload = request.data
            sig = request.headers.get('Stripe-Signature')

            event = stripe.Webhook.construct_event(
                payload, sig, os.getenv('STRIPE_WEBHOOK_SECRET')
            )

            if event['type'] == 'checkout.session.completed':
                session = event['data']['object']
                tenant_id = session.metadata['tenant_id']

                # Activate subscription
                subscription =
Subscription.query.filter_by(tenant_id=tenant_id).first()
                subscription.status = 'active'
                subscription.stripe_customer_id = session.customer
                subscription.stripe_subscription_id = session.subscription

                tenant = Tenant.query.get(tenant_id)
                tenant.status = 'active'

                db.session.commit()

            return jsonify({'success': True})
```

USER FLOWS

1. Tenant Signup (Self-Service)

```
/signup/tenant
├─ Form: company_name, subdomain, contact_email, contact_name,
password
├─ Create tenant (status=trial, trial_ends_at=+14days)
├─ Create admin user (role=admin)
├─ Send welcome email
└─ Redirect: https://{subdomain}.lex-cao.replit.app/login
```

2. Regular User Login & Chat

```
/{subdomain}.lex-cao.replit.app/login
├─ Login with email + password
├─ Middleware: Extract tenant from subdomain
├─ Check: tenant.status in ['trial', 'active']
└─ Redirect: /chat

/chat
├─ Sidebar: List of user's chats
├─ Main area: Chat messages
├─ Input: Type question for LEX
└─ On send:
    ├─ Show LEX animation
    ├─ Call Vertex AI agent
    ├─ Hide animation
    └─ Show LEX response
```

3. Tenant Admin Dashboard

```
/admin/dashboard
├─ Stats: Total users, chats, questions asked
├─ Links:
│   ├─ /admin/users (add/remove users)
│   ├─ /admin/billing (view subscription, upgrade)
│   └─ /admin/analytics (usage charts)
```

4. Super Admin Dashboard

```
/super-admin/dashboard
├─ List all tenants
├─ Stats: MRR, total users, active/trial/suspended
├─ Actions:
│   ├─ Create tenant (manual)
│   ├─ Activate/suspend tenant
│   └─ Impersonate tenant (login as their admin)
```

PAGES TO BUILD

Public (No Auth)

- `/` - Landing page with pricing
- `/login` - Login form
- `/signup/tenant` - Tenant self-service signup

User (Auth Required)

- `/chat` - Main chat interface with LEX
- `/profile` - Edit profile

Tenant Admin (Auth + Role=Admin)

- `/admin/dashboard` - Tenant overview
- `/admin/users` - User management (add/remove/deactivate)
- `/admin/billing` - Subscription management
- `/admin/analytics` - Usage statistics

Super Admin (Auth + SuperAdmin)

- `/super-admin/dashboard` - Master control panel
- `/super-admin/tenants` - All tenants list
- `/super-admin/analytics` - Global metrics

LANDING PAGE CONTENT

Hero Section:

```
🤖 LEX CAO Expert

Altijd het juiste antwoord op je CAO vragen

LEX is jouw AI-assistent met kennis van 70+ CAO documenten.
Perfect voor uitzendbureaus.

[14 Dagen Gratis Proberen →]
```

Features: - 📚 70+ CAO Documenten (Glastuinbouw, ABU, NBBU, etc) - ⚡ Instant Antwoorden (geen zoeken, direct het juiste artikel) - 👥 Team Collaboration (heel je payroll team) - 🔒 100% Privacy (jouw data blijft van jouw bedrijf)

Pricing:

```
Professional          Enterprise
€499/maand           €1.199/maand

✓ 5 gebruikers       ✓ Unlimited gebruikers
✓ Unlimited vragen   ✓ Unlimited vragen
✓ Chat history       ✓ White-label branding
✓ Priority support   ✓ Custom domain
✓ PDF exports        ✓ Dedicated support

[Start Trial]        [Start Trial]
```

EMAIL TEMPLATES

Welcome Email (New User):

```
Subject: Welkom bij LEX CAO Expert!

Hoi {first_name},

Je account is aangemaakt voor {company_name}.

Login: https://{subdomain}.lex-cao.nl

Veel succes met LEX! 🤖
```

Trial Expiring (3 days before):

```
Subject: Je LEX trial loopt over 3 dagen af

Hoi {contact_name},

Je 14-daagse trial loopt bijna af. Upgrade nu om door te gaan.

[Upgrade Naar Professional →]
```

Payment Failed:

```
Subject: ⚠️ Betaling mislukt

Hoi {contact_name},

We konden je betaling niet verwerken. Update je betaalmethode om
actief te blijven.

[Betaalmethode Updaten →]
```

SECURITY REQUIREMENTS

1. **Password hashing:** Use bcrypt (via werkzeug.security)
2. **JWT tokens:** For API authentication
3. **CSRF protection:** Flask-WTF
4. **SQL injection prevention:** Use SQLAlchemy ORM (no raw queries)
5. **XSS prevention:** Escape all user input
6. **Rate limiting:** 100 requests/minute per user
7. **Audit logging:** Log all admin actions (who did what when)

TESTING CHECKLIST

After building, test:

1. ✅ Tenant signup works → creates tenant + admin user
2. ✅ Subdomain routing works → `xyz.lex-cao.replit.app` loads correct tenant
3. ✅ Login works → redirects to /chat
4. ✅ Chat with LEX works → gets response from Vertex AI
5. ✅ LEX animation shows → random scenario plays during wait
6. ✅ Chat history saves → messages persist in database
7. ✅ "New chat" button works → creates new chat
8. ✅ Tenant admin can add users → max 5 for Professional
9. ✅ Tenant data isolation → Tenant A cannot see Tenant B's chats
10. ✅ Trial expires after 14 days → blocks access + shows upgrade prompt
11. ✅ Stripe checkout works → redirects to payment
12. ✅ Webhook activates subscription → status changes to 'active'
13. ✅ Super admin can see all tenants → lists in dashboard
14. ✅ Email notifications work → welcome email arrives

DEPLOYMENT SETTINGS

In Replit: 1. Enable "Always On" deployment 2. Set custom domain: `lex-cao.nl` (with wildcard `*.lex-cao.nl`) 3. Environment: Production 4. Auto-deploy: On push to main

FILE STRUCTURE

```
lex-cao-expert/
├── main.py                      # Flask entry point
├── requirements.txt             # Dependencies
│
├── backend/
│   ├── __init__.py
│   ├── app.py                  # App factory
│   ├── config.py               # Config
│   ├── extensions.py           # DB, login manager, etc
│   │
│   ├── models/
│   │   ├── super_admin.py
│   │   ├── tenant.py
│   │   ├── user.py
│   │   ├── chat.py
│   │   ├── message.py
│   │   └── subscription.py
│   │
│   ├── routes/
│   │   ├── auth.py           # Login/register
│   │   ├── chat.py           # Chat endpoints
│   │   ├── tenant_admin.py   # Tenant admin panel
│   │   ├── super_admin.py    # Super admin panel
│   │   └── webhooks.py       # Stripe webhooks
│   │
│   ├── services/
│   │   ├── lex_agent.py      # Vertex AI integration
│   │   ├── stripe_service.py # Payment processing
│   │   └── email_service.py  # SendGrid emails
│   │
│   └── middleware/
│       └── tenant_context.py # Tenant isolation
│
├── frontend/
│   ├── static/
│   │   ├── css/
│   │   │   └── style.css
│   │   └── js/
│   │       ├── chat.js
│   │       └── lex-animation.js
│   │
│   └── templates/
│       ├── base.html
│       ├── landing.html
│       ├── login.html
│       ├── chat.html
│       ├── tenant_admin_dashboard.html
│       └── super_admin_dashboard.html
```

DEPENDENCIES (requirements.txt)

```
Flask==3.0.0
Flask-SQLAlchemy==3.1.1
Flask-Login==0.6.3
Flask-Migrate==4.0.5
Flask-CORS==4.0.0
psycopg2-binary==2.9.9
google-cloud-aiplatform==1.38.1
vertexai==1.38.1
stripe==7.8.0
sendgrid==6.11.0
bcrypt==4.1.2
PyJWT==2.8.0
python-dotenv==1.0.0
```

SUCCESS CRITERIA

Product is complete when:

✅ Landing page is live and professional
✅ Tenant can self-signup (no manual intervention)
✅ Trial works (14 days free, no credit card)
✅ Tenant admin can add 5 users (Professional plan)
✅ Users can chat with LEX via Vertex AI
✅ LEX animation shows random scenarios during wait
✅ Chat history persists per user
✅ Stripe payment flow works (€499/€1.199)
✅ After payment: tenant status = 'active'
✅ Trial expiration blocks access + shows upgrade
✅ Tenant data is 100% isolated (no cross-tenant leaks)
✅ Super admin can view/manage all tenants
✅ Email notifications work (welcome, trial expiring, payment failed)
✅ Subdomain routing works (`xyz.lex-cao.replit.app`)
✅ Mobile responsive

IMPORTANT NOTES

1. **LEX agent already works** - Just call Vertex AI API, don't rebuild RAG
2. **70 documents already uploaded** - Don't upload documents again
3. **Use Flask-Login** for session management (easier than JWT for this use case)
4. **Keep it simple** - Use server-side rendering (Jinja templates) + minimal JavaScript
5. **Tailwind CSS via CDN** - No build process needed
6. **Test subdomain routing locally** - Use `127.0.0.1:5000` with different query params to simulate subdomains

ANTI-PATTERNS TO AVOID

❌ Don't rebuild RAG - use existing Vertex AI agent
❌ Don't use React/Vue - keep it simple with vanilla JS
❌ Don't write complex auth - use Flask-Login
❌ Don't forget tenant_id in queries - causes data leaks
❌ Don't hardcode API keys - use environment variables
❌ Don't skip email verification - causes spam signups
❌ Don't allow unlimited trial - enforce 14 days

BUILD ORDER

1. **Database models** - Define all models first
2. **Auth system** - Login/register/logout
3. **Tenant isolation middleware** - Critical for security
4. **Chat with LEX** - Core functionality
5. **LEX animation** - Makes waiting fun
6. **Tenant admin panel** - User management
7. **Super admin panel** - Your control center
8. **Stripe integration** - Payments
9. **Email notifications** - Communication
10. **Landing page** - Marketing

FINAL INSTRUCTION

Build the complete, production-ready application with all features listed above. Make it professional, secure, and ready to onboard paying customers immediately after deployment.

Focus on: - Clean, maintainable code - Proper error handling - User-friendly UI - Mobile responsive design - Fast loading times - Intuitive navigation

This is a real business - treat it like a professional SaaS product.