Allice Platform - Enhanced Features Documentation

Overview

This document describes the enhanced features added to the Allice platform for Pro tier users. The enhancements include:

- **API Gateway with FastAPI**: Modern REST API with OpenAPI documentation
- **User Authentication**: JWT-based authentication with role-based access control
- **Capabilities Management**: Dynamic capability system with rate limiting
- **Application Registry**: Track and manage deployed applications
- **AI Model Integrations**: Replicate and Gemini API support
- **Social Media Integration**: Twitter and LinkedIn posting
- **Cloud Management**: Deploy to AWS, GCP, and DigitalOcean
- **Admin Dashboard**: Web-based administration interface
- **Ollama Tunnel**: Connect to Ollama on your Mac via Twingate

---

Table of Contents

Architecture

Technology Stack

- **API Framework**: FastAPI (Python)
- **Database**: PostgreSQL with SQLAlchemy ORM
- **Authentication**: JWT with python-jose
- **Rate Limiting**: Custom middleware with database tracking
- **Frontend**: Vanilla JavaScript with modern CSS
- **Deployment**: Docker + Google Cloud Run

Project Structure

```
viralspark_ailice/
├── api/                          # API package
│   ├── models.py                 # Database models
│   ├── database.py               # Database connection
│   ├── auth.py                   # Authentication utilities
│   ├── capabilities.py           # Capabilities management
│   ├── rate_limiter.py           # Rate limiting middleware
│   ├── schemas.py                # Pydantic schemas
│   ├── routers/                  # API routes
│   │   ├── auth.py               # Authentication endpoints
│   │   ├── applications.py       # Application registry
│   │   ├── scraping.py           # Web scraping
│   │   ├── social.py             # Social media
│   │   ├── cloud.py              # Cloud management
│   │   ├── ai_models.py          # AI model APIs
│   │   └── admin.py              # Admin endpoints
│   └── integrations/             # External service integrations
│       ├── scraper.py            # Web scraping logic
│       ├── social.py             # Social media APIs
│       ├── cloud.py              # Cloud deployment
│       ├── replicate_api.py      # Replicate integration
│       └── gemini_api.py         # Gemini integration
├── static/                       # Static files
│   └── admin_dashboard.html      # Admin dashboard UI
├── fastapi_app.py                # FastAPI application
├── capabilities_config.json      # Capabilities configuration
├── config.json                   # AIlice configuration
├── Dockerfile                    # Docker image
└── .env.example                  # Environment variables template
```

Getting Started

Prerequisites

- Python 3.10+
- PostgreSQL 12+
- Docker (for deployment)

Local Setup

1. **Clone the repository** (if not already done):
   ```bash
   cd /home/ubuntu/viralspark_ailice
   ```

2. **Create virtual environment**:
   ```bash
   python3 -m venv venv
   source venv/bin/activate
   ```

3. **Install dependencies**:
   ```bash
   pip install -r requirements.txt
   ```

4. **Set up environment variables**:
   `bash`
   `cp .env.example .env`
   `# Edit .env with your settings`

5. **Set up PostgreSQL database**:
   ```bash
   # Create database
   createdb ailice

# Or using psql
psql -U postgres -c “CREATE DATABASE ailice;”
```

1. **Initialize database**:
   `bash`
   `# The database will be initialized automatically on first run`
   `# Or manually:`
   `python -c "from api.database import init_db; init_db()"`

2. **Run the application**:
   ```bash
   # Using FastAPI (recommended)
   python fastapi_app.py

# Or using uvicorn directly
uvicorn fastapi_app:app –host 0.0.0.0 –port 8080 –reload
```

1. **Access the application**:
   - API Documentation: http://localhost:8080/docs
   - Admin Dashboard: http://localhost:8080/admin/dashboard
   - Health Check: http://localhost:8080/health

Authentication

User Registration

Endpoint: `POST /api/auth/register`

Request:

```
{
  "username": "johndoe",
  "email": "john@example.com",
  "password": "securepassword123"
}
```

Response:

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "token_type": "bearer",
  "user": {
    "id": 1,
    "username": "johndoe",
    "email": "john@example.com",
    "role": "user",
    "is_active": true,
    "created_at": "2025-12-22T00:00:00Z"
  }
}
```

User Login

Endpoint: `POST /api/auth/login`

Request:

```
{
  "username": "johndoe",
  "password": "securepassword123"
}
```

Response: Same as registration

Using JWT Token

Include the token in the `Authorization` header:

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc..." \
     http://localhost:8080/api/auth/me
```

Role-Based Access Control

Two roles are supported:
- **user**: Regular users with access to standard features
- **admin**: Full access including cloud management and admin dashboard

---

Capabilities System

Overview

The capabilities system provides fine-grained control over what features are available and enforces rate limits.

Configuration

Capabilities are configured in `capabilities_config.json`:

```
{
  "capabilities": {
    "web_scraping": {
      "enabled": true,
      "permissions": ["read", "write"],
      "rate_limit": "100/hour",
      "endpoints": ["/api/scrape", "/api/browse"],
      "description": "Web browsing, scraping, and automation"
    },
    "social_media": {
      "enabled": true,
      "platforms": ["twitter", "linkedin"],
      "rate_limit": "50/hour",
      "endpoints": ["/api/social/post", "/api/social/schedule"],
      "description": "Social media posting and management"
    },
    "cloud_management": {
      "enabled": false,
      "requires_admin": true,
      "providers": ["aws", "gcp", "digitalocean"],
      "endpoints": ["/api/cloud/deploy", "/api/cloud/manage"],
      "description": "Cloud infrastructure management (admin only)"
    }
  }
}
```

Rate Limiting

Rate limits are enforced per user per capability:

- **100/hour**: Web scraping and browsing
- **50/hour**: Social media posting
- **Custom**: Can be configured per capability

Rate limit tracking is stored in the database and resets based on the configured window (hour/day/minute).

Enabling/Disabling Capabilities

Capabilities can be toggled via:

1. **Admin Dashboard**: Toggle switches in the UI
2. **API**: `PUT /api/admin/capabilities/{capability_name}`
3. **Configuration File**: Edit `capabilities_config.json` and restart

API Endpoints

Authentication

- `POST /api/auth/register` - Register new user
- `POST /api/auth/login` - Login user
- `GET /api/auth/me` - Get current user info

Applications

- `POST /api/applications` - Create application

- `GET /api/applications` - List applications
- `GET /api/applications/{id}` - Get application details
- `PUT /api/applications/{id}` - Update application
- `DELETE /api/applications/{id}` - Delete application

Web Scraping

- `POST /api/scrape` - Scrape a URL
- `POST /api/browse` - Browse website interactively

Social Media

- `POST /api/social/post` - Post to social media
- `POST /api/social/schedule` - Schedule a post

Cloud Management (Admin Only)

- `POST /api/cloud/deploy` - Deploy application
- `GET /api/cloud/manage` - List deployments

AI Models

- `POST /api/ai/replicate` - Call Replicate API
- `POST /api/ai/gemini` - Call Gemini API
- `GET /api/ai/models` - List available models

Admin (Admin Only)

- `GET /api/admin/stats` - Get system statistics
- `GET /api/admin/capabilities` - Get capabilities configuration
- `PUT /api/admin/capabilities/{name}` - Update capability
- `GET /api/admin/users` - List all users
- `PUT /api/admin/users/{id}/role` - Update user role
- `PUT /api/admin/users/{id}/status` - Update user status

Admin Dashboard

Accessing the Dashboard

Navigate to: `http://your-domain/admin/dashboard`

Note: You must be logged in with an admin account to access the dashboard.

Features

1. **System Statistics**:
   - Total users
   - Active users
   - Total applications
   - Deployed applications
   - API calls today

2. **Capabilities Management**:
   - View all capabilities

   - Enable/disable capabilities
   - View rate limits and configuration

3. **User Management**:
   - View all users
   - See user roles and status
   - Creation dates

4. **Applications**:
   - View all deployed applications
   - Application status
   - Subdomain information

Authentication

The dashboard stores the JWT token in `localStorage`. If not authenticated, you'll be redirected to `/login`.

To implement a login page, create a simple HTML form that posts to `/api/auth/login`.

AI Model Integrations

Replicate API

Setup:

```
# Set environment variable
export REPLICATE_API_TOKEN=your_token_here
```

Usage:

```
curl -X POST http://localhost:8080/api/ai/replicate \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "stability-ai/sdxl",
    "prompt": "A beautiful sunset over mountains",
    "parameters": {
      "width": 1024,
      "height": 1024
    }
  }'
```

Gemini API

Setup:

```
# Set environment variable
export GEMINI_API_KEY=your_key_here
```

Usage:

```
curl -X POST http://localhost:8080/api/ai/gemini \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "gemini-pro",
    "prompt": "Explain quantum computing in simple terms",
    "parameters": {
      "temperature": 0.7,
      "max_output_tokens": 1024
    }
  }'
```

Ollama Tunnel Setup

For detailed instructions on setting up Ollama tunnel via Twingate, see ollama_tunnel_config.md (./ollama_tunnel_config.md).

Quick Start

1. **Install Twingate on your Mac**:
   ```
   bash
     brew install --cask twingate
   ```

2. **Configure Ollama to accept remote connections**:
   ```
   bash
     export OLLAMA_HOST=0.0.0.0:11434
     ollama serve
   ```

3. **Get your Twingate IP**:
   ```
   bash
     ifconfig | grep -A 1 utun
   ```

4. **Update Allice config**:
   ```
   json
     {
       "models": {
         "ollama": {
           "baseURL": "http://100.64.1.25:11434/v1"
         }
       }
     }
   ```

Deployment

Docker Build

```
# Build image
docker build -t ailice-platform .

# Run container
docker run -p 8080:8080 \
  -e DATABASE_URL=postgresql://user:pass@host/db \
  -e JWT_SECRET_KEY=your_secret_key \
  ailice-platform
```

Google Cloud Run

```
# Build and push to Google Container Registry
gcloud builds submit --tag gcr.io/YOUR_PROJECT_ID/ailice-platform

# Deploy to Cloud Run
gcloud run deploy ailice-platform \
  --image gcr.io/YOUR_PROJECT_ID/ailice-platform \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated \
  --set-env-vars DATABASE_URL=your_db_url,JWT_SECRET_KEY=your_secret
```

Environment Variables

Required:
- `DATABASE_URL`: PostgreSQL connection string
- `JWT_SECRET_KEY`: Secret key for JWT tokens

Optional:
- `PORT`: Server port (default: 8080)
- `REPLICATE_API_TOKEN`: Replicate API token
- `GEMINI_API_KEY`: Gemini API key
- `TWITTER_API_KEY`: Twitter API credentials
- `LINKEDIN_ACCESS_TOKEN`: LinkedIn access token

Security Best Practices

1. **Change JWT Secret**:
   `bash`
   `# Generate a secure secret`
   `python -c "import secrets; print(secrets.token_urlsafe(32))"`

2. **Use HTTPS**: Always use HTTPS in production

3. **Rotate API Keys**: Regularly rotate all API keys

4. **Database Security**:
   - Use strong passwords

   - Enable SSL for database connections
   - Restrict network access

5. **Rate Limiting**: Monitor and adjust rate limits based on usage
6. **CORS Configuration**: Restrict CORS origins in production
7. **Admin Access**: Limit admin accounts to trusted users
8. **Logging**: Enable comprehensive logging for security audits

---

Support and Troubleshooting

Common Issues

1. **Database Connection Errors**:
   - Check `DATABASE_URL` format
   - Ensure PostgreSQL is running
   - Verify network connectivity
2. **Authentication Failures**:
   - Check JWT_SECRET_KEY is set
   - Verify token hasn't expired
   - Ensure user is active
3. **Rate Limit Exceeded**:
   - Check usage in admin dashboard
   - Wait for rate limit window to reset
   - Contact admin to increase limits

Logs

Application logs are written to stdout/stderr. In Cloud Run, view logs:

```
gcloud run logs read ailice-platform --limit 100
```

Getting Help

- Check documentation: README.md (./README.md)
- View API docs: http://your-domain/docs
- Contact: support@viralspark.ai

Changelog

Version 1.0.0 (2025-12-22)

Added:
- FastAPI-based API gateway
- JWT authentication system
- Role-based access control
- Capabilities management with rate limiting

- Application registry
- Web scraping endpoints
- Social media integration
- Cloud management endpoints
- Replicate API integration
- Gemini API integration
- Admin dashboard UI
- Ollama tunnel configuration
- Comprehensive documentation

Changed:
- Updated Dockerfile for FastAPI
- Enhanced environment variables
- Improved security with JWT

Fixed:
- Database connection pooling
- Error handling in endpoints

License

See LICENSE (./LICENSE) file for details.

Contributing

We welcome contributions! Please:

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Submit a pull request

Last Updated: December 22, 2025