ViralSpark Allice Enhancements - Implementation Summary

Overview

This document summarizes the comprehensive enhancements made to the Allice platform, transforming it into an enterprise-grade API platform with payment processing, media handling, and extensive integrations.

Implementation Date

December 22, 2024

Key Features Implemented

1. Stripe Payment Integration ✅

Files Created/Modified:
- `api/stripe_service.py` - Complete Stripe service module
- `api/routers/billing.py` - Billing and payment endpoints
- `.env.example` - Added Stripe configuration variables

Features:
- Product ID: `prod_TYtmG0y2uNXjSU`
- Price: $49.99/month (`price_1Sblz7LZxEDQErW5uQyWN5F3`)
- Subscription creation and management
- Checkout session handling
- Webhook processing for payment events
- Invoice retrieval
- Subscription cancellation/reactivation

Endpoints:
- `POST /api/billing/checkout` - Create checkout session
- `GET /api/billing/subscription` - Get subscription details
- `POST /api/billing/subscription/cancel` - Cancel subscription
- `POST /api/billing/subscription/reactivate` - Reactivate subscription
- `GET /api/billing/invoices` - List invoices
- `POST /api/billing/webhook` - Stripe webhook handler

2. Database Models Expansion ✅

File Modified:
- `api/models.py` - Added 10 new models

New Models:
1. **Subscription** - Track Stripe subscriptions
2. **Item** - Generic content/data items (CRUD)
3. **MediaFile** - Video, audio, image files
4. **Notification** - User notifications

5. **ChatMessage** - Collaboration chat messages
6. **FileUpload** - General file uploads
7. **IntegrationStatus** - External service integrations
8. **AnalyticsEvent** - Usage and performance tracking
9. **AIModel** - AI/ML model registry
10. **SharedResource** - Resource sharing for collaboration

3. API Endpoint Categories (12 Total) ✅

A. Authentication & User Management

File: `api/routers/auth.py` (existing, enhanced)
- User registration
- Login
- Get current user
- Logout

B. Content/Data CRUD

File: `api/routers/items.py` (NEW)
- `GET /api/items` - List items
- `POST /api/items` - Create item
- `GET /api/items/{id}` - Get item
- `PUT /api/items/{id}` - Update item
- `DELETE /api/items/{id}` - Delete item

C. Media Handling with FFmpeg

File: `api/routers/media.py` (NEW)
- `POST /api/media/upload` - Upload media
- `GET /api/media/{id}/stream` - Stream media
- `POST /api/media/{id}/transcode` - Transcode media
- FFmpeg integration for video/audio processing
- Automatic metadata extraction

D. Analytics & Metrics

File: `api/routers/analytics.py` (NEW)
- `GET /api/analytics/usage` - Usage statistics
- `GET /api/analytics/errors` - Error tracking
- `GET /api/analytics/performance` - Performance metrics

E. Notifications

File: `api/routers/notifications.py` (NEW)
- `POST /api/notify/send` - Send notification
- `GET /api/notify/history` - Notification history
- `POST /api/notify/{id}/read` - Mark as read
- `POST /api/notify/mark-all-read` - Mark all as read

F. Payments & Billing

File: `api/routers/billing.py` (NEW)
- See Stripe Integration section above

G. Search & Discovery

File: `api/routers/search.py` (NEW)
- `GET /api/search` - Full-text search
- `GET /api/recommendations` - Personalized recommendations

H. AI/ML Inference

File: `api/routers/ai_inference.py` (NEW)
- `POST /api/ai/predict` - Run inference
- `POST /api/ai/train` - Train model (placeholder)
- `GET /api/ai/models` - List models
- Support for OpenAI, Google Gemini, Replicate

I. Collaboration

File: `api/routers/collaboration.py` (NEW)
- `POST /api/chat/send` - Send chat message
- `GET /api/chat/{room}` - Get chat messages
- `POST /api/collab/share` - Share resource
- `GET /api/collab/shared` - List shared resources

J. System & Admin

File: `api/routers/system.py` (NEW)
- `GET /api/system/health` - Health check
- `GET /api/system/stats` - System statistics
- `GET /api/system/config` - Configuration
- `POST /api/system/restart` - Restart system
- `GET /api/system/logs` - View logs

K. File Management

File: `api/routers/files.py` (NEW)
- `POST /api/files/upload` - Upload file
- `GET /api/files` - List files
- `GET /api/files/{id}/download` - Download file
- `DELETE /api/files/{id}` - Delete file

L. External Integrations

File: `api/routers/integrations.py` (NEW)
- `POST /api/integrations/{service}/connect` - Connect integration
- `POST /api/integrations/{service}/sync` - Sync data
- `GET /api/integrations/{service}/status` - Get status
- `GET /api/integrations` - List all integrations
- `DELETE /api/integrations/{service}` - Disconnect integration

Supported Services:
- Google Drive
- TradingView
- Dropbox
- OneDrive
- GitHub
- GitLab
- Slack

- Discord
- Twitter
- LinkedIn

4. LLM Optimization ✅

File Modified:
- `requirements.txt` - Removed heavy ML dependencies

Removed Packages (~5GB reduction):
- torch
- transformers
- accelerate
- bitsandbytes
- peft
- datasets
- tensorboard
- gpt4all
- huggingface_hub
- einops
- librosa
- sounddevice
- ChatTTS
- marker-pdf

Added Packages:
- stripe>=7.0.0
- ffmpeg-python>=0.2.0
- redis>=5.0.0
- websockets>=12.0
- python-email-validator>=2.1.0
- celery>=5.3.0
- PyPDF2>=3.0.0

Benefits:
- Significantly smaller Docker image
- Faster deployment
- Lower resource requirements
- API-based LLM access only (no local GPU needed)

5. Docker & Infrastructure Updates ✅

File Modified:
- `Dockerfile`

Changes:
- Added Redis server
- Enhanced FFmpeg comment
- Optimized layer caching
- Updated Python dependency installation notes
- Removed references to heavy model installations

6. Environment Configuration ✅

File Modified:
- `.env.example`

New Variables:
- Stripe configuration (5 variables)
- Redis URL
- Email/SMTP settings
- External integration API keys
- JWT secret
- File upload configuration
- FFmpeg timeout

7. Application Updates ✅

File Modified:
- `fastapi_app.py`

Changes:
- Imported 11 new routers
- Organized router inclusion by category
- Added clear comments for router organization

8. Documentation ✅

Files Created:
- `API_ENDPOINTS.md` - Comprehensive API documentation (300+ lines)
- `README.md` (updated) - Added ViralSpark enhancements section
- `VIRALSPARK_ENHANCEMENTS.md` (this file)

Documentation Includes:
- All 12 endpoint categories
- Request/response examples
- Stripe setup guide
- Quick start examples
- Authentication guide
- Error handling documentation

File Structure

```
/home/ubuntu/viralspark_ailice/
├── api/
│   ├── models.py (MODIFIED - added 10 new models)
│   ├── stripe_service.py (NEW - Stripe integration)
│   └── routers/
│       ├── ai_inference.py (NEW)
│       ├── analytics.py (NEW)
│       ├── billing.py (NEW)
│       ├── collaboration.py (NEW)
│       ├── files.py (NEW)
│       ├── integrations.py (NEW)
│       ├── items.py (NEW)
│       ├── media.py (NEW)
│       ├── notifications.py (NEW)
│       ├── search.py (NEW)
│       └── system.py (NEW)
├── .env.example (MODIFIED)
├── Dockerfile (MODIFIED)
├── requirements.txt (MODIFIED)
├── fastapi_app.py (MODIFIED)
├── API_ENDPOINTS.md (NEW)
├── README.md (MODIFIED)
└── VIRALSPARK_ENHANCEMENTS.md (NEW)
```

Statistics

- **New Router Files:** 11
- **New Database Models:** 10
- **New API Endpoints:** ~50+
- **Lines of Code Added:** ~3,000+
- **Documentation Pages:** 3 (300+ lines total)
- **Docker Image Size Reduction:** ~5GB
- **Supported External Services:** 10+

Testing Checklist

Prerequisites

- [ ] PostgreSQL database running
- [ ] Redis server running
- [ ] Stripe API keys configured
- [ ] Environment variables set

Endpoint Tests

- [ ] Authentication (register, login, me)
- [ ] Stripe checkout creation
- [ ] Stripe webhook processing
- [ ] Items CRUD operations
- [ ] Media upload and streaming
- [ ] File upload and download

- [ ] Analytics queries
- [ ] Notifications
- [ ] Search functionality
- [ ] AI inference with OpenAI/Google
- [ ] Chat messaging
- [ ] Resource sharing
- [ ] System health check
- [ ] Integration connections

Integration Tests

- [ ] End-to-end subscription flow
- [ ] Media transcoding pipeline
- [ ] Real-time chat
- [ ] External service sync (Google Drive, etc.)

Deployment Instructions

1. Local Development

```
# Copy environment file
cp .env.example .env

# Edit .env with your configurations
nano .env

# Install dependencies
pip install -r requirements.txt

# Run migrations (if using Alembic)
alembic upgrade head

# Start the application
python3 fastapi_app.py
```

2. Docker Deployment

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

# Run with docker-compose
docker-compose up -d
```

3. Production Deployment

See `DEPLOYMENT_GUIDE.md` for detailed production deployment instructions.

Configuration

Required Environment Variables

```
# Database
DATABASE_URL=postgresql://ailice:password@localhost:5432/ailice_db

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

# Redis
REDIS_URL=redis://localhost:6379/0

# JWT
JWT_SECRET_KEY=your_secret_key_here
```

Optional Environment Variables

See `.env.example` for the complete list of optional configurations.

API Usage Examples

1. Complete Subscription Flow

```
# Step 1: Register
TOKEN=$(curl -X POST http://localhost:8080/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{"username":"user","email":"user@example.com","password":"pass123"}' \
  | jq -r '.access_token')

# Step 2: Create checkout
curl -X POST http://localhost:8080/api/billing/checkout \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"success_url":"http://localhost:8080/success","cancel_url":"http://localhost:
8080/cancel"}'

# Step 3: After payment, check subscription
curl -X GET http://localhost:8080/api/billing/subscription \
  -H "Authorization: Bearer $TOKEN"
```

2. Media Processing

```
# Upload video
MEDIA_ID=$(curl -X POST http://localhost:8080/api/media/upload \
  -H "Authorization: Bearer $TOKEN" \
  -F "file=@video.mp4" \
  | jq -r '.id')

# Transcode
curl -X POST http://localhost:8080/api/media/$MEDIA_ID/transcode \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"output_format":"mp4","quality":"high"}'

# Stream
curl -X GET http://localhost:8080/api/media/$MEDIA_ID/stream \
  -H "Authorization: Bearer $TOKEN"
```

3. AI Inference

```
# OpenAI
curl -X POST http://localhost:8080/api/ai/predict \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "model_id":"openai:gpt-4",
    "input_data":{"prompt":"Explain quantum computing"}
  }'

# Google Gemini
curl -X POST http://localhost:8080/api/ai/predict \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "model_id":"google:gemini-pro",
    "input_data":{"prompt":"Write a Python function"}
  }'
```

Known Limitations

1. **Model Training**: The `/api/ai/train` endpoint is a placeholder and requires custom implementation
2. **WebSocket**: Real-time chat currently uses polling; WebSocket implementation pending
3. **Email**: SMTP configuration required for email notifications
4. **Background Tasks**: Celery configuration required for async task processing

Future Enhancements

- [ ] WebSocket implementation for real-time features
- [ ] Email notification system
- [ ] Background job queue (Celery)
- [ ] Advanced analytics dashboards
- [ ] Machine learning model training infrastructure
- [ ] Video streaming optimization (HLS/DASH)

- [ ] Advanced search with Elasticsearch
- [ ] Multi-tenant support
- [ ] API versioning
- [ ] Rate limiting per subscription tier

Support & Contact

For issues, questions, or contributions:
- GitHub Issues: AIlice Repository (https://github.com/djNaughtyb/AIlice)
- Documentation: See `API_ENDPOINTS.md`
- Interactive Docs: `/docs` endpoint

License

This enhanced version maintains the same license as the original AIlice project.

Implementation Completed: December 22, 2024
Version: 1.0.0 (ViralSpark Enhanced Edition)
Developer: Enhanced for ViralSpark deployment