# ServeFlow AI

## Architecture & Progress Documentation

## 1. Project Overview

ServeFlow AI is a modern, AI-powered service marketplace platform that connects customers with service providers through intelligent request analysis, real-time communication, and transparent financial workflows.

The system is designed as a **production-ready, event-driven platform** rather than a prototype, with emphasis on:

* Resilient AI integrations
* Real-time UX
* Multi-role system design
* Scalable backend architecture

## 2. Technology Stack

### Backend

* **Framework**: Django REST Framework (DRF)
* **Real-Time Layer**: Django Channels + Redis
* **Database**: SQLite3 (current), schema designed for PostgreSQL migration
* **AI Engine**: Google Gemini Pro Vision (google-generativeai SDK)
* **Authentication**: DRF Token Authentication (HTTP + WebSocket)
* **Email**: SMTP
* **Async & Events**: Signals + WebSocket event bus

### Frontend

* **Framework**: React.js (Vite)
* **Styling**: Tailwind CSS + Custom Glassmorphism System
* **Animations**: Framer Motion
* **State Management**: Context API
* **Routing**: React Router DOM
* **HTTP Client**: Axios
* **Real-Time**: Native WebSocket integration via Context

## 3. Major Architectural Upgrades (Completed)

### 3.1 Production-Grade AI Intelligence Engine

**Objective**  
Replace mock or static AI responses with a resilient, production-capable AI system.

**Implementation**

* Integrated **Google Gemini Pro Vision**
* Backend module:  
  backend/api/views\_ai.py

**Key Capabilities**

* **Visual Request Analysis**
  + Accepts image uploads (multipart / base64)
  + Sends image + contextual prompt to Gemini Vision
  + Returns structured JSON:
    - Category
    - Technical title
    - Description
    - Urgency level
    - Estimated budget
* **Multi-Key Resilience System**
  + SystemSettings model stores **multiple independent API keys**
  + Automatic key rotation on 429 rate-limit errors
* **Graceful Degradation**
  + If all AI keys fail, system falls back to simulated AI output
  + Prevents request failure and preserves UX continuity

**Status**: ✅ Completed & Production-Ready

### 3.2 Real-Time Event Bus (WebSocket Ecosystem)

**Objective**  
Eliminate HTTP polling and move to a fully event-driven architecture.

**Infrastructure**

* Django Channels
* Redis (channels\_redis) as channel layer
* Supports fan-out and scalable real-time messaging

**Security**

* Custom TokenAuthMiddleware
* DRF token validation for WebSocket connections

**Communication Channels**

* ws/notifications/
  + Global notifications
  + Job offers
  + Status updates
  + System alerts
* ws/chat/<job\_id>/
  + Isolated provider–customer chat rooms
  + Message persistence

**Frontend Integration**

* WebSocketContext.jsx
  + Centralized connection handling
  + Auto-reconnect strategy
  + Global state dispatch

**Status**: ✅ Completed

### 3.3 UI / UX Overhaul (Glassmorphism System)

**Objective**  
Upgrade from a functional UI to a premium, modern product experience.

**Global Design System**

* Implemented in index.css
* Core utilities:
  + .glass, .glass-card
  + Backdrop blur
  + Semi-transparent layered panels

**Animation System**

* Custom motion patterns:
  + blob (floating gradients)
  + scale-in (modal & card transitions)

**Component Modernization**

* Landing Page:
  + 3D hero section
  + Animated feature cards
* Dashboards:
  + Dashboard.jsx
  + ProviderDashboard.jsx
  + AdminPanel.jsx
* Navigation:
  + Scroll-aware transparent navbar

**Status**: ✅ Completed

## 4. Backend Core Modules

|  |  |  |
| --- | --- | --- |
| **Module** | **Purpose** | **Status** |
| Authentication | User / Provider / Admin roles | ✅ Complete |
| Request Engine | Service lifecycle management | ✅ Complete |
| AI Service | Request & dispute analysis | ✅ Complete |
| Matching System | Bid-based provider matching | ✅ Complete |
| Financials | Invoices & commissions | ⚠ Partial |
| Messaging | Real-time chat | ✅ Complete |
| Notifications | Email + UI alerts | ✅ Complete |
| Audit Logs | Security & traceability | ✅ Complete |

## 5. Frontend Feature Modules

### 5.1 Mission Control – Provider Dashboard

**File**

* frontend/src/pages/ProviderDashboard.jsx

**Core Concepts**

* Jobs treated as **“Missions”**
* State machine:
  + Pending → Initiated → Completed
* Embedded real-time chat
* Live statistics engine:
  + Trust rating
  + Revenue
  + Mission success rate

**Status**: ✅ Completed

### 5.2 Intelligent Admin Control Panel

**Files**

* AdminSettings.jsx
* AdminCategories.jsx

**Key Features**

* Live theme & color preview
* Service category management with image previews
* Secure AI key vault (multi-key management)

**Status**: ✅ Completed

### 5.3 AI-Assisted Visual Request Creation

**File**

* frontend/src/pages/CreateRequest.jsx

**Workflow**

1. User uploads an image
2. AI analyzes the issue
3. System auto-fills:
   * Title
   * Description
   * Category
4. Backend maps AI output to internal category IDs

**Impact**

* Reduced user friction
* Higher-quality requests
* Faster provider matching

**Status**: ✅ Completed

## 6. Recent Sprint: Stability & Bug Fixes

**Delivered Fixes**

* Auth-aware navbar logic
* Branding & SEO consistency
* Footer icon import issues
* Admin save-flow repairs
* Image preview normalization
* Provider dashboard crash prevention
* Profile image uploads via FormData
* Notification triggers verified

**Status**: ✅ Completed

## 7. Gap Analysis (Current)

|  |  |  |
| --- | --- | --- |
| **Feature** | **Status** | **Notes** |
| Core Job Loop | Completed | End-to-end operational |
| Real-Time Chat | Completed | Persistent + secure |
| Visual Design | Completed | Glassmorphism system |
| Payments | Pending | Stripe integration planned |
| Dockerization | Planned | Dockerfile + compose |

## 8. Overall Project Status

**ServeFlow AI has moved beyond prototype stage.**  
The platform now has:

* A **real AI engine**
* **Event-driven real-time UX**
* **Multi-role dashboards**
* **Production-grade architecture decisions**

Remaining work is **infrastructure hardening**, not core system design.

**DIAGRAMS**

Providers

Create-Request

Services

Login / Register

**SCREENSHOTS**