BridgeBot AI Frontend - Multi-Model AI Interface

A professional, developer-focused web application for interacting with multiple AI models, managing knowledge bases, and streamlining AI workflows.

BridgeBot AI Interface

🚀 Quick Start

Prerequisites

- Node.js 18+ and npm/pnpm
- Python 3.8+ (for BridgeBot backend, if used)

Installation

1. **Clone the repository**
   `bash git clone <repository-url> cd bridgebot-ui`
2. **Install dependencies**
   ```bash
   # Frontend
   cd frontend
   pnpm install

# Backend
cd ../backend
   ```

npm install
```

1. **Configure environment**
   ```bash
   # Copy environment templates
   cp .env.example .env
   cp backend/.env.example backend/.env

# Edit .env files with your API keys
```

1. **Start the application**
   ```bash
   # Terminal 1 - Backend
   cd backend
   npm run dev

# Terminal 2 - Frontend
cd frontend
pnpm dev
```

1. **Access the application**
   - Frontend: http://localhost:5173
   - Backend API: http://localhost:3001

🔧 Configuration

API Keys Setup

Add your API keys to `backend/.env`:
```

```
# Required for AI model access
OPENAI_API_KEY=sk-your-openai-key
CLAUDE_API_KEY=your-claude-key
GEMINI_API_KEY=your-gemini-key
BRIDGEBOT_TOKEN=your-bridgebot-token
BRIDGEBOT_ENDPOINT_URL=https://your-bridgebot-instance.com
```

Supported AI Models

Provider	Models	Status
OpenAI	GPT-4o, GPT-4, GPT-3.5 Turbo	✅ Ready
Claude	Claude 3 Opus, Sonnet, Haiku	✅ Ready
Gemini	Gemini Pro, Pro Vision	✅ Ready
BridgeBot	BridgeBot Tutor, General	✅ Ready

🎯 Features

🤖 Multi-Model Chat

- **Seamless Model Switching**: Switch between OpenAI, Claude, Gemini, and BridgeBot
- **Unified Interface**: Consistent experience across all models
- **Real-time Health Monitoring**: Live status of API connections
- **Advanced Parameters**: Temperature, max tokens, and model-specific settings

📚 Knowledge Base Management

- **Document Upload**: Drag & drop PDF, TXT, MD, DOC, DOCX files
- **Full-Text Search**: Advanced search across all uploaded documents

- **Smart Chunking**: Intelligent text segmentation for better context
- **Category Organization**: Organize documents by project or topic

📋 Prompt Template System

- **Built-in Templates**: Pre-configured templates for common tasks
- Code Review
- Concept Explanation
- Debug Help
- Document Summarization
- API Documentation
- Meeting Notes
- **Custom Templates**: Create and share your own templates
- **Variable Support**: Dynamic placeholders for reusable prompts
- **Template Rendering**: Preview and customize before use

⚡ Command System

- **@bridgebot run** - Execute BridgeBot-specific workflows
- **@gpt** - Send commands to OpenAI models
- **@claude** - Send commands to Claude models
- **@kb search** - Search knowledge base
- **@template** - Load prompt templates
- **@help** - Show all available commands

🔐 Authentication & Session Management

- **User Accounts**: Simple username/password authentication
- **Demo Mode**: Try the interface without creating an account
- **Secure API Key Storage**: Encrypted storage of API credentials

- **Session Persistence**: Stay logged in across browser sessions

🏗️ Architecture

Three-Panel Layout

```
┌─────────────────────────────────────────────────────────────┐
│                      Header Bar                             │
├──────────────┬──────────────────────────────┬───────────────┤
│              │                              │               │
│ Left Panel   │     Main Chat Area           │  Right Panel  │
│              │                              │               │
│ • Models     │ • Message History            │ • Knowledge   │
│ • Config     │ • Chat Input                 │ • Templates   │
│ • Settings   │ • Commands                   │ • File Upload │
│ • Status     │ • Markdown Rendering         │ • Search      │
│              │                              │               │
└──────────────┴──────────────────────────────┴───────────────┘
```

Backend API Structure

```
/api
├── /auth          # User authentication
├── /models        # AI model management
├── /chat          # Chat operations
├── /knowledge     # Document management
├── /templates     # Prompt templates
└── /health        # System health
```

📖 Usage Guide

Getting Started

1. **Login or Demo**: Choose to create an account or use demo mode
2. **Configure Models**: Add API keys in the left sidebar
3. **Start Chatting**: Use the main chat area to interact with AI
4. **Upload Documents**: Drag files to the right panel to build knowledge base
5. **Use Commands**: Type @ to access advanced commands

Model Selection

1. **Open Left Sidebar**: Click the panel button in the header
2. **Expand Provider**: Click on OpenAI, Claude, Gemini, or BridgeBot
3. **Add API Key**: Enter your API key for the provider
4. **Select Model**: Choose from available models
5. **Configure Settings**: Adjust temperature and token limits

Knowledge Base Usage

1. **Upload Documents**: Drag & drop files to the right panel
2. **Wait for Processing**: Documents are automatically processed
3. **Search Content**: Use the search box to find information
4. **Use in Chat**: Reference documents with @kb search commands

Template Workflow

1. **Browse Templates**: Check the right panel for available templates
2. **Select Template**: Click on a template to view details
3. **Use Command**: Type @template in chat
4. **Fill Variables**: Provide required template variables

5. **Generate Content**: Template renders with your inputs

Command Examples

```
# Search knowledge base
@kb search authentication methods

# Use a template
@template code-review

# BridgeBot specific workflow
@bridgebot run test_paper

# Get help
@help
```

🔌 API Integration

Adding New AI Models

1. **Update Models Config** (`backend/src/routes/models.js`):
   `javascript const MODELS_CONFIG = { 'new-provider': { id: 'new-provider', name: 'New Provider', models: [/* model definitions */], endpoint: 'https://api.newprovider.com', requiresKey: true } };`
2. **Implement API Service** (`backend/src/services/apiService.js`):
   `javascript async function sendNewProviderRequest(modelName, message, options) { // Implementation for new provider }`
3. **Add Health Check**:
   `javascript async function checkNewProviderHealth(config) { // Health check implementation }`

Custom Prompt Templates

Create templates in the backend or via the API:

```
{
  "name": "Custom Template",
  "category": "development",
  "description": "My custom template",
  "prompt": "Please {action} the following {content}...",
  "variables": ["action", "content"]
}
```

🛠️ Development

Project Structure

```
bridgebot-ui/
├── frontend/               # React frontend
│   ├── src/
│   │   ├── components/    # UI components
│   │   ├── stores/        # Zustand state management
│   │   ├── services/      # API services
│   │   └── utils/         # Utilities
├── backend/               # Express backend
│   ├── src/
│   │   ├── routes/        # API routes
│   │   ├── services/      # Business logic
│   │   └── middleware/    # Express middleware
├── shared/                # Shared types/utilities
├── docs/                  # Documentation
└── prompts/               # Example templates
```

Available Scripts

Frontend:

```
pnpm dev          # Development server
pnpm build        # Production build
pnpm preview      # Preview build
pnpm lint         # Lint code
```

Backend:

```
npm run dev       # Development server with nodemon
npm start         # Production server
npm test          # Run tests
```

Environment Variables

Variable	Description	Required
`VITE_API_URL`	Backend API URL	Yes
`OPENAI_API_KEY`	OpenAI API key	Optional
`CLAUDE_API_KEY`	Claude API key	Optional
`GEMINI_API_KEY`	Gemini API key	Optional
`BRIDGEBOT_TOKEN`	BridgeBot token	Optional

🚀 Deployment

Docker Deployment

```
# Build and run with Docker Compose
docker-compose up -d

# Or build manually
docker build -t bridgebot-frontend ./frontend
docker build -t bridgebot-backend ./backend
```

Manual Deployment

1. **Build Frontend**:
   `bash cd frontend pnpm build`
2. **Deploy Backend**:
   `bash cd backend npm start`
3. **Serve Frontend**: Use nginx, Apache, or any static file server

Environment Setup

- **Production**: Set `NODE_ENV=production`
- **SSL/TLS**: Configure HTTPS for production
- **Database**: Consider using PostgreSQL/MongoDB for production
- **Monitoring**: Add logging and monitoring solutions

🤝 Contributing

1. **Fork the repository**
2. **Create feature branch**: `git checkout -b feature/amazing-feature`

3. **Commit changes**: `git commit -m 'Add amazing feature'`
4. **Push to branch**: `git push origin feature/amazing-feature`
5. **Open Pull Request**

Development Guidelines

- **TypeScript**: Use TypeScript for type safety
- **ESLint**: Follow linting rules
- **Components**: Create reusable UI components
- **Testing**: Write tests for new features
- **Documentation**: Update docs for API changes

📝 License

This project is licensed under the MIT License - see the LICENSE file for details.

🆘 Support

- **Issues**: Report bugs via GitHub Issues
- **Discussions**: Join GitHub Discussions for questions
- **Documentation**: Check the `/docs` folder for detailed guides
- **API Reference**: See `/docs/api.md` for complete API documentation

🔮 Roadmap

- [ ] **Real-time Streaming**: WebSocket support for streaming responses
- [ ] **Multi-user Support**: Team workspaces and collaboration
- [ ] **Advanced RAG**: Vector embeddings and semantic search
- [ ] **Plugin System**: Custom integrations and extensions

- [ ] **Mobile App**: Native mobile applications
- [ ] **Self-hosting**: One-click deployment solutions

Built with ❤️ for developers and AI enthusiasts