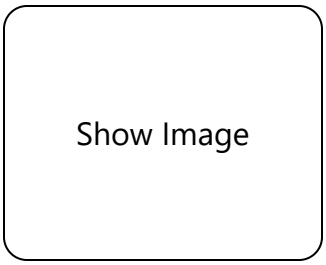
LegalEase - AI Legal Assistant Platform

A comprehensive legal assistant platform that provides AI-powered legal guidance, contract generation, and case research capabilities for the Indian legal system.



🌟 Features

- **AI Legal Chat**: Get instant legal advice powered by Google Gemini AI
- **Contract Generator**: Create professional contracts with 6+ templates (Rental, Employment, Service, Sales, NDA, Partnership)
- **Case Research**: Search through Indian legal database and precedents
- **User Management**: Secure authentication with password reset functionality
- **Admin Dashboard**: Complete admin panel for user and contract management
- **Document Export**: Generate PDF contracts and save for future reference
- **Real-time Search**: Integration with Indian Kanoon API for live legal case data

🚀 Tech Stack

Backend

- **Node.js** with Express.js
- **SQLite3** database with SQL.js
- **JWT** authentication
- **bcrypt** password hashing
- **nodemailer** for email services

Frontend

- **Vanilla JavaScript** with modern ES6+
- **CSS3** with glassmorphism design
- **Responsive design** for all devices

AI & Search

- **Python FastAPI** RAG service
- **Google Gemini AI** for legal guidance
- **Indian Kanoon API** integration
- **FAISS** for vector search (optional)

📋 Prerequisites

Before running this application, make sure you have:

- **Node.js** (v18 or higher)
- **Python** (v3.8 or higher)
- **npm** (v9 or higher)
- **Git**

🛠️ Installation & Setup

1. Clone the Repository

```bash
git clone https://github.com/yourusername/legalease.git
cd legalease
```

2. Backend Setup

```bash
# Install Node.js dependencies
npm install

# Or if you have a package-lock.json
npm ci
```

3. Python RAG Service Setup

```bash
```

```
# Install Python dependencies
pip install -r requirements.txt

# Or using virtual environment (recommended)
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r requirements.txt
```

4. Environment Configuration

Create a `.env` file in the project root:

```
env

# JWT Configuration
JWT_SECRET=your_super_secret_jwt_key_here

# Email Configuration (Gmail)
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-gmail-app-password

# Google AI Configuration
GOOGLE_API_KEY=your-google-gemini-api-key

# Indian Kanoon API (Optional)
INDIAN_KANOON_API_KEY=your-indian-kanoon-api-key

# Server Configuration
PORT=5000
NODE_ENV=development
```

5. Database Initialization

The SQLite database will be automatically created when you first run the server. It includes:

- Users table with admin account
- Contracts table for document storage
- Contract templates with pre-built forms
- Password reset tokens table

🚀 Running the Application

Start the Backend Server

```bash
npm start
# Or for development with auto-reload
npm run dev
```

Start the RAG Service

In a separate terminal:

```bash
python rag_service.py
```

Or using uvicorn:

```bash
uvicorn rag_service:app --host 0.0.0.0 --port 8000 --reload
```

Access the Application

- **Main Application**: http://localhost:5000
- **RAG Service API**: http://localhost:8000
- **RAG Service Docs**: http://localhost:8000/docs

🕵️ Default Admin Account

```
Email: adminsid@gmail.com
Password: Coffee@030903
```

📂 Project Structure

```
legalease/
├── server.js              # Main Express server
├── rag_service.py         # Python FastAPI service for AI
├── package.json           # Node.js dependencies
├── requirements.txt       # Python dependencies
```

```
├── legalease.db          # SQLite database (auto-created)
├── .env                  # Environment variables
├── .gitignore            # Git ignore rules
├── README.md             # Project documentation
│
├── Frontend Files:
├── login.html            # Login page
├── signup.html           # Registration page
├── index.html            # Main dashboard
├── contracts.html        # Contract generator
├── research.html         # Case research
├── chat.html             # AI chat interface
├── profile.html          # User profile
├── admin-dashboard.html  # Admin panel
├── style.css             # Main stylesheet
└── script.js             # Frontend JavaScript
```

🔗 API Endpoints

Authentication

- `POST /api/signup` - User registration
- `POST /api/login` - User login
- `POST /api/forgot-password` - Request password reset
- `POST /api/verify-reset-code` - Verify reset code
- `POST /api/reset-password` - Reset password

User Management

- `GET /api/profile` - Get user profile
- `POST /api/change-password` - Change password

Contracts

- `GET /api/contract-templates` - Get all templates
- `GET /api/contracts` - Get user contracts
- `POST /api/contracts` - Create new contract
- `PUT /api/contracts/:id` - Update contract
- `DELETE /api/contracts/:id` - Delete contract
- `GET /api/contracts/analytics` - Contract analytics

AI & Search

- `POST /api/chat` - AI legal chat
- `GET /api/search` - Case research

Admin Routes

- `GET /api/admin/users` - Manage users
- `GET /api/admin/stats` - Platform statistics
- `GET /api/admin/contracts` - All contracts

🎨 Contract Templates

The platform includes 6 professional contract templates:

1. **Rental Agreement** - Comprehensive residential property rental
2. **Employment Contract** - Standard employment with Indian labor law compliance
3. **Service Agreement** - Professional service contracts
4. **Sales Agreement** - Goods and services sales contracts
5. **Non-Disclosure Agreement (NDA)** - Confidentiality agreements
6. **Partnership Agreement** - Business partnership with profit sharing

🔐 Security Features

- JWT-based authentication
- Bcrypt password hashing
- SQL injection protection
- CORS enabled
- Input validation and sanitization
- Secure password reset with email verification
- Admin role-based access control

🌐 Deployment

Quick Deploy Options

1. Railway (Recommended)

```bash
```

```
# Push to GitHub first
git add .
git commit -m "Initial commit"
git push origin main

# Deploy on Railway
1. Go to railway.app
2. Connect GitHub repo
3. Add environment variables
4. Deploy automatically
```

2. Render

bash

```
# Similar to Railway
1. Go to render.com
2. Connect GitHub repo
3. Configure build/start commands
4. Add environment variables
```

3. Vercel (Frontend + Serverless)

bash

```
# Install Vercel CLI
npm i -g vercel

# Deploy
vercel --prod
```

Environment Variables for Production

Make sure to set these in your deployment platform:

env

```
NODE_ENV=production
JWT_SECRET=your-production-jwt-secret
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-gmail-app-password
GOOGLE_API_KEY=your-google-gemini-api-key
INDIAN_KANOON_API_KEY=your-indian-kanoon-api-key
RAG_SERVICE_URL=https://your-rag-service-url.com
```

📧 Email Configuration

Gmail Setup

1. Enable 2-Factor Authentication
2. Generate an App Password:
   - Go to Google Account settings
   - Security → App passwords
   - Generate password for "Mail"
3. Use this App Password in EMAIL_PASS

🔑 API Keys Setup

Google Gemini AI

1. Go to Google AI Studio
2. Create new API key
3. Add to GOOGLE_API_KEY environment variable

Indian Kanoon API (Optional)

1. Visit Indian Kanoon
2. Request API access
3. Add key to INDIAN_KANOON_API_KEY

🐛 Troubleshooting

Common Issues

1. **Database Connection Error**

```
bash
```

```
# Check if database file exists and has proper permissions
ls -la legalease.db
```

2. **RAG Service Connection Failed**

```bash
# Ensure Python service is running on port 8000
curl http://localhost:8000/health
```

3. **Email Service Failed**

```bash
# Check email configuration in logs
# Ensure Gmail App Password is correct
```

4. **Module Not Found Errors**

```bash
# Reinstall dependencies
rm -rf node_modules package-lock.json
npm install
```

Development Mode Issues

```bash
# Clear database and restart fresh
rm legalease.db
npm start

# Check all services are running
curl http://localhost:5000/api/profile
curl http://localhost:8000/health
```

🤝 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

📄 License

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

👨‍💻 Author

LegalEase Team

- Email: support@legalease.ai
- GitHub: @yourusername

🙏 Acknowledgments

- Google Gemini AI for legal guidance capabilities
- Indian Kanoon for legal database access
- Open source community for various libraries used

📊 Project Stats

- **Languages**: JavaScript, Python, HTML, CSS
- **Framework**: Node.js, Express, FastAPI
- **Database**: SQLite3
- **AI**: Google Gemini, RAG Architecture
- **Deployment**: Railway/Render Ready

🔮 Future Enhancements

- [ ] Multi-language support (Hindi, Tamil, etc.)
- [ ] Voice-to-text legal queries
- [ ] Advanced contract analytics
- [ ] Integration with more legal databases
- [ ] Mobile app development
- [ ] Blockchain document verification
- [ ] Advanced AI legal reasoning

⚖️ **Disclaimer**: LegalEase is an AI assistant tool. Always consult qualified legal professionals for official legal advice.