Full-Stack Web Application Template

A modern, production-ready full-stack web application template built with React, FastAPI, and MongoDB. This template provides a solid foundation for building scalable web applications with authentication, CRUD operations, and a responsive user interface.

🚀 Features

- **Frontend**: React 18 with Vite, TailwindCSS, and shadcn/ui components
- **Backend**: FastAPI with Python, JWT authentication, and async operations
- **Database**: MongoDB with Motor (async driver)
- **Authentication**: JWT-based authentication with protected routes
- **Admin Dashboard**: Full CRUD operations for content management
- **Real-time Updates**: Changes in dashboard reflect immediately on public pages
- **Responsive Design**: Mobile-first design with TailwindCSS
- **Type Safety**: TypeScript-ready with Pydantic models
- **Production Ready**: Environment-based configuration and deployment setup

📋 Prerequisites

Before you begin, ensure you have the following installed:

- **Node.js** (v18 or higher)
- **Python** (v3.11 or higher)
- **MongoDB** (v5.0 or higher) or MongoDB Atlas account
- **pnpm** (recommended) or npm

🛠️ Quick Start

1. Clone the Repository

Bash

```bash
git clone <repository-url>
cd fullstack-template
```

2. Backend Setup

Bash

```bash
# Navigate to backend directory
cd backend

# Create virtual environment (optional but recommended)
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Create environment file
cp .env.example .env

# Edit .env file with your configuration
# MONGODB_URL=mongodb://localhost:27017
# JWT_SECRET_KEY=your-super-secret-jwt-key
```

3. Frontend Setup

Bash

```bash
# Navigate to frontend directory
cd ../frontend

# Install dependencies
pnpm install  # or npm install

# Create environment file
cp .env.example .env

# Edit .env file with your configuration
# VITE_API_URL=http://localhost:8000
```

4. Start the Application

Terminal 1 - Backend:

Bash

```bash
cd backend
python main.py
```

Terminal 2 - Frontend:

Bash

```bash
cd frontend
pnpm run dev --host  # or npm run dev -- --host
```

The application will be available at:

- Frontend: http://localhost:5173
- Backend API: http://localhost:8000
- API Documentation: http://localhost:8000/docs

📁 Project Structure

Plain Text

```
fullstack-template/
├── backend/                  # FastAPI backend
│   ├── main.py              # Main application file
│   ├── main_test.py         # Test version with in-memory storage
│   ├── requirements.txt     # Python dependencies
│   ├── .env.example        # Environment variables template
│   └── .env                # Environment variables (create from example)
├── frontend/                # React frontend
│   ├── src/
│   │   ├── components/     # Reusable components
│   │   │   ├── layout/    # Layout components (Navbar, Footer)
│   │   │   ├── ui/        # shadcn/ui components
│   │   │   └── ProtectedRoute.jsx
│   │   ├── contexts/      # React contexts
│   │   │   └── AuthContext.jsx
│   │   ├── lib/           # Utility functions
│   │   │   └── api.js     # API client
│   │   ├── pages/         # Page components
│   │   │   ├── Home.jsx
│   │   │   ├── About.jsx
│   │   │   ├── Contact.jsx
│   │   │   ├── Login.jsx
│   │   │   ├── Signup.jsx
│   │   │   └── Dashboard.jsx
│   │   ├── App.jsx        # Main app component
│   │   └── main.jsx       # Entry point
│   ├── package.json       # Node.js dependencies
│   ├── .env.example      # Environment variables template
│   ├── .env              # Environment variables (create from example)
│   └── vite.config.js    # Vite configuration
```

```
├── .gitignore           # Git ignore file
└── README.md           # This file
```

🔧 Configuration

Backend Environment Variables

Create a `.env` file in the `backend` directory:

Plain Text

```
# MongoDB Configuration
MONGODB_URL=mongodb://localhost:27017
DATABASE_NAME=fullstack_template

# JWT Configuration
JWT_SECRET_KEY=your-super-secret-jwt-key-change-this-in-production

# API Configuration
FRONTEND_URL=http://localhost:5173
BACKEND_URL=http://localhost:8000
```

Frontend Environment Variables

Create a `.env` file in the `frontend` directory:

Plain Text

```
# API Configuration
VITE_API_URL=http://localhost:8000

# Frontend Configuration
VITE_APP_NAME=FullStack Template
VITE_APP_VERSION=1.0.0
```

📖 Usage Guide

Authentication

The application includes a complete authentication system:

1. **Sign Up**: Create a new account at `/signup`
2. **Login**: Access your account at `/login`

3. **Protected Routes**: Dashboard is only accessible to authenticated users
4. **JWT Tokens**: Secure token-based authentication with automatic refresh

Demo Credentials

For testing purposes, use these credentials:

- **Username**: demo
- **Password**: demo123

Pages Overview

- **Home** (`/`): Public landing page with latest posts
- **About** (`/about`): Information about the template and technologies
- **Contact** (`/contact`): Contact form and information
- **Login** (`/login`): User authentication
- **Sign Up** (`/signup`): User registration
- **Dashboard** (`/dashboard`): Protected admin area for CRUD operations

CRUD Operations

The dashboard provides full CRUD functionality:

1. **Create**: Add new posts with title, content, and category
2. **Read**: View all your posts with pagination
3. **Update**: Edit existing posts
4. **Delete**: Remove posts with confirmation

🔌 API Endpoints

Authentication Endpoints

Plain Text

```
POST /auth/signup
POST /auth/login
GET /auth/me
```

Posts Endpoints

Plain Text

```
GET /posts              # Get all posts (public)
POST /posts             # Create new post (protected)
GET /posts/{id}         # Get specific post
PUT /posts/{id}         # Update post (protected, owner only)
DELETE /posts/{id}      # Delete post (protected, owner only)
```

API Documentation

Visit `http://localhost:8000/docs` for interactive API documentation powered by FastAPI's automatic OpenAPI generation.

🎨 Frontend Components

Layout Components

- **Navbar**: Responsive navigation with authentication state
- **Footer**: Site footer with links and information
- **Layout**: Main layout wrapper for all pages

UI Components

The template uses shadcn/ui components for a consistent design system:

- Buttons, Cards, Forms, Dialogs
- Input fields, Textareas, Select dropdowns
- Alerts, Badges, Loading states
- Responsive grid and layout utilities

Authentication Context

The `AuthContext` provides:

- User state management
- Login/logout functionality
- Protected route handling
- Token management

🚀 Deployment

Backend Deployment

1. **Environment Setup**:
2. **Install Dependencies**:
3. **Run Application**:

Frontend Deployment

1. **Build for Production**:
2. **Serve Static Files**:
   The `dist` folder contains the built application ready for deployment to any static hosting service.

Database Setup

Local MongoDB

Bash

```
# Install MongoDB
# Ubuntu/Debian
sudo apt-get install mongodb

# macOS with Homebrew
brew install mongodb-community

# Start MongoDB service
sudo systemctl start mongodb  # Linux
brew services start mongodb-community  # macOS
```

MongoDB Atlas (Cloud)

1. Create account at MongoDB Atlas
2. Create a new cluster
3. Get connection string
4. Update `MONGODB_URL` in your `.env` file

🧪 Testing

Backend Testing

Bash

```
cd backend
python main_test.py  # Runs with in-memory storage for testing
```

Frontend Testing

Bash

```
cd frontend
pnpm run test  # or npm run test
```

🔒 Security Considerations

Production Checklist

- [ ] Change default JWT secret key
- [ ] Use environment variables for sensitive data
- [ ] Enable HTTPS in production
- [ ] Configure CORS for specific origins
- [ ] Set up proper MongoDB authentication
- [ ] Implement rate limiting
- [ ] Add input validation and sanitization
- [ ] Set up logging and monitoring

Environment Variables

Never commit `.env` files to version control. Use `.env.example` as a template and create your own `.env` files locally.

🎯 Customization

Adding New Pages

1. Create a new component in `frontend/src/pages/`
2. Add route in `frontend/src/App.jsx`
3. Update navigation in `frontend/src/components/layout/Navbar.jsx`

Adding New API Endpoints

1. Add new route in `backend/main.py`
2. Create Pydantic models for request/response
3. Update API client in `frontend/src/lib/api.js`

Styling Customization

The template uses TailwindCSS for styling:

- Modify `frontend/tailwind.config.js` for theme customization
- Update component styles in individual files
- Use shadcn/ui components for consistency

Database Schema

To add new collections or modify existing ones:

1. Update Pydantic models in `backend/main.py`
2. Add new database operations
3. Update frontend components and API calls

🛠️ Development Tips

Hot Reload

Both frontend and backend support hot reload:

- Frontend: Vite automatically reloads on file changes
- Backend: Use `uvicorn main:app --reload` for auto-restart

Debugging

- **Frontend**: Use browser developer tools and React DevTools
- **Backend**: FastAPI provides detailed error messages and `/docs` endpoint
- **Database**: Use MongoDB Compass for visual database management

Code Organization

- Keep components small and focused
- Use TypeScript for better type safety

- Follow REST API conventions
- Implement proper error handling

🐛 Troubleshooting

Common Issues

Backend Issues

MongoDB Connection Error

```
Plain Text
pymongo.errors.ServerSelectionTimeoutError
```

- Ensure MongoDB is running
- Check connection string in `.env`
- Verify network connectivity

JWT Token Error

```
Plain Text
Could not validate credentials
```

- Check JWT secret key configuration
- Verify token expiration settings
- Clear browser localStorage and re-login

Frontend Issues

API Connection Error

```
Plain Text
Network Error or CORS Error
```

- Ensure backend is running on correct port
- Check `VITE_API_URL` in frontend `.env`
- Verify CORS configuration in backend

Build Errors

```
Module not found or Import errors
```

- Run `pnpm install` to ensure all dependencies are installed
- Check import paths and component names
- Clear node_modules and reinstall if needed

Performance Optimization

- **Frontend**: Use React.memo for expensive components
- **Backend**: Implement database indexing for frequently queried fields
- **Database**: Use aggregation pipelines for complex queries
- **Caching**: Implement Redis for session storage and caching

📚 Learning Resources

Technologies Used

- **React**: Official Documentation
- **FastAPI**: Official Documentation
- **MongoDB**: Official Documentation
- **TailwindCSS**: Official Documentation
- **Vite**: Official Documentation

Tutorials and Guides

- React Router Tutorial
- FastAPI Tutorial
- MongoDB University
- TailwindCSS Components

🤝 Contributing

We welcome contributions! Please follow these steps:

1. Fork the repository

2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

Development Guidelines

- Follow existing code style and conventions
- Add tests for new features
- Update documentation for any changes
- Ensure all tests pass before submitting PR

📄 License

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

🙏 Acknowledgments

- shadcn/ui for the beautiful UI components
- Lucide for the icon library
- FastAPI for the amazing Python web framework
- React for the powerful frontend library

📞 Support

If you have any questions or need help:

1. Check the Issues page
2. Read the documentation thoroughly
3. Search for existing solutions online
4. Create a new issue with detailed information

Happy coding! 🚀

Built with ❤️ for the developer community.