**Chatbot System Technical Documentation**

**Project Overview**

This document outlines the technical specifications for developing a customizable chatbot system with a React frontend and Python FastAPI backend. The system will allow users to create, customize, and deploy AI-powered chatbots on their websites through an embeddable widget.

**Timeline**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Task** | **Duration (in days)** |
| Planning & Design | Requirements gathering, UI mockups, architecture | 2 |
| Frontend Development | Front Page, Login Screen, user dashboard, chat widget, embeddable JS snippet | 10 |
| Backend Development | APIs for data ingestion (URL/file/text), chatbot model training, context management | 14 |
| Customization & Logic | Logo upload, tone config, persona, initial messages | 5 |
| Admin Panel | Admin login, chatbot analytics, user management, content moderation | 10 |
| Embedding | Iframe widget and JS snippet | 6 |
| Testing & QA | Comprehensive testing of all functionalities | 7 |
| Deployment | Setting up production environment and deployment pipelines | 5 |
| **Total days** |  | **59 days** |
| **Total Months** |  | **3 Months** |

**System Architecture**

**High-Level Architecture**

┌────────────────┐ ┌───────────────────┐ ┌─────────────────┐

│ React Frontend │◄────►│ FastAPI Backend │◄────►│ Database (MYSQL/ │

│ - User Portal │ │ - Authentication │ │ NoSQL) │

│ - Admin Panel │ │ - API Endpoints │ └─────────────────┘

│ - Chat Widget │ │ - Business Logic │ ▲

└─────────────────┘ └───────────────────┘ │

▲ │

│ │

▼ ▼

┌───────────────────┐ ┌─────────────────┐

│ AI Model Service │ │ File/Document │

│ - Training │ │ Storage │

│ - Inference │ │ │

└───────────────────┘ └─────────────────┘

**Frontend Architecture (React)**

* **Component-Based Structure**: Utilize React's component architecture for modularity
* **State Management**: Redux or Context API for global state management
* **Routing**: React Router for navigation between pages
* **UI Framework**: Material UI or Tailwind CSS
* **API Integration**: Axios for RESTful API calls

**Backend Architecture (Python FastAPI)**

* **API Framework**: FastAPI for high-performance RESTful APIs
* **Authentication**: JWT-based authentication
* **Database Access**: SQLAlchemy ORM or direct DB drivers
* **AI Integration**: Integration with machine learning models
* **File Handling**: For document uploads and processing

**Database**

* **User Database**: Mysql for structured user and configuration data
* **Chat History**: Mysql for flexible chat history storage
* **Vector Database**: For semantic search and embeddings (optional: Pinecone or Chromadb or Faiss index)

**Technical Requirements**

**Frontend Requirements (React)**

**1. Landing Page**

* Marketing content and feature showcase
* Registration and login buttons
* Product demos and examples
* Pricing information

**2. Authentication System**

* User registration form
* Login form with password recovery
* Social login options (Google, GitHub)
* Session management

**3. User Dashboard**

* Overview of chatbot configurations
* Usage statistics and analytics
* Quick access to edit existing chatbots
* Create new chatbot wizard

**4. Chatbot Customization Interface**

* Visual editor for chatbot appearance
* Persona and tone configuration
* Knowledge base management
* Training data upload interface

**5. Chat Widget**

* Real-time chat interface
* User message history
* Typing indicators
* File upload capability
* Responsive design for mobile and desktop

**6. Embeddable Components**

* JavaScript snippet generator
* iFrame generator
* Configuration options for embedding

**7. Admin Panel**

* User management
* Content moderation tools
* System-wide analytics
* Configuration settings

**Backend Requirements (Python FastAPI)**

**1. Authentication API**

* User registration endpoint
* Login authentication
* Password reset functionality
* JWT token management
* Role-based access control

**2. Chatbot Management API**

* CRUD operations for chatbot configurations
* Settings persistence
* Version control for chatbot configurations

**3. Data Ingestion API**

* URL scraping endpoint
* File upload handlers (PDF, DOCX, TXT, CSV)
* Text input processing
* Data transformation and cleaning

**4. Knowledge Base API**

* Vector storage integration
* Document chunking and processing
* Semantic search capabilities
* Context window management

**5. Chat API**

* Message handling
* Context tracking
* Session management
* Rate limiting

**6. Analytics API**

* Usage statistics collection
* Performance metrics
* User engagement tracking
* Error logging

**7. Admin API**

* User management endpoints
* System configuration
* Content moderation tools

**AI Model Integration**

**1. Model Training**

* Fine-tuning pipeline
* Training data preprocessing
* Model evaluation metrics

**2. Inference Engine**

* Real-time inference optimization
* Caching mechanisms
* Response generation
* Context handling

**3. Knowledge Retrieval**

* Vector search implementation
* Relevance scoring
* Source citation

**Implementation Plan**

**Week 1: Foundation & Setup**

* Environment setup and project initialization
* Database schema creation
* Authentication system implementation
* Basic API structure
* Frontend project structure setup

**Week 2: Core Features Development**

* User dashboard development
* Chatbot configuration interfaces
* Basic chat functionality
* Data ingestion API endpoints
* Initial AI model integration

**Week 3: Advanced Features & Integration**

* Knowledge base system completion
* Advanced chatbot customization
* Embeddable widget development
* Analytics system implementation
* Extended AI capabilities

**Week 4: Admin Panel & Testing**

* Admin panel development
* User management system
* Content moderation tools
* Comprehensive testing
* Bug fixes and improvements

**Week 5: Deployment & Finalization**

* Production environment setup
* Deployment pipeline configuration
* Performance optimization
* Documentation completion
* User training materials

**Technical Stack**

**Frontend**

* **Framework**: React 18
* **State Management**: Redux Toolkit or Context API
* **Routing**: React Router 6
* **UI Components**: Material UI or Tailwind CSS
* **HTTP Client**: Axios
* **Form Handling**: Formik or React Hook Form
* **Testing**: Jest, React Testing Library

**Backend**

* **Framework**: FastAPI
* **Authentication**: JWT tokens with Pydantic validation
* **ORM**: SQLAlchemy
* **Database Drivers**: psycopg2 for PostgreSQL, Motor for MongoDB
* **Task Queue**: Celery with Redis (for background processing)
* **File Processing**: PyPDF2, python-docx, BeautifulSoup
* **Vector Embeddings**: sentence-transformers, spaCy

**DevOps & Infrastructure**

* **Containerization**: Docker
* **Orchestration**: Docker Compose (development), Kubernetes (production)
* **CI/CD**: GitHub Actions or GitLab CI
* **Monitoring**: Prometheus, Grafana
* **Logging**: ELK Stack or Loki

**Security Considerations**

* Implement proper authentication and authorization
* Secure data storage with encryption at rest
* Rate limiting to prevent abuse
* Input validation and sanitization
* Regular security audits
* GDPR compliance for user data
* Content filtering for inappropriate material

**Performance Considerations**

* Database indexing for frequently accessed data
* Caching strategies (Redis)
* Pagination for large datasets
* Asynchronous processing for resource-intensive tasks
* Load balancing for production deployment
* CDN integration for static assets
* Optimized AI model inference

**Testing Strategy**

* Unit tests for individual components
* Integration tests for API endpoints
* End-to-end tests for critical user flows
* Performance testing under load
* Security vulnerability testing
* Cross-browser compatibility testing
* Mobile responsiveness testing

**Deployment Strategy**

* Development environment: Docker Compose
* Staging environment: Kubernetes on cloud provider
* Production environment: Kubernetes on cloud provider with replicas
* Blue-green deployment for zero-downtime updates
* Automated backups
* Disaster recovery plan