# **High-Level Design Document (HLD)**

## **ChatApp: Enterprise AI Chat Application**

### **Document Information**

* **Project Name**: ChatApp
* **Document Type**: High-Level Design (HLD)
* **Version**: 1.0

## **1. Introduction**

### **1.1 Purpose**

This High-Level Design document provides detailed technical specifications for the ChatApp-LAB45 enterprise AI chat application, describing the system's functional components, interfaces, and design decisions.

### **1.2 Scope**

The document covers:

* Detailed functional component design
* Technical implementation specifications
* API and interface designs
* Security and performance considerations
* Integration patterns and data flows

### **1.3 Document Conventions**

* **Components**: Capitalized (e.g., ChatInterface)
* **APIs**: REST endpoints (e.g., /api/completions)
* **Technologies**: Framework names (e.g., Next.js, React)
* **Data**: JSON schema definitions

## **2. System Overview**

### **2.1 System Context Diagram**

┌─────────────────────────────────────┐  
 │ Internet │  
 └─────────────────┬───────────────────┘  
 │  
 ┌─────────────────┴───────────────────┐  
 │ Edge Network │  
 │ (Vercel/Azure CDN) │  
 └─────────────────┬───────────────────┘  
 │  
 ┌─────────────────────────────────┴─────────────────────────────────┐  
 │ ChatApp-LAB45 System │  
 │ │  
 │ ┌─────────────────┐ ┌─────────────────┐ │  
 │ │ Frontend │◄────────┤ Backend API │ │  
 │ │ (Next.js) │ │ (Next.js) │ │  
 │ │ │ │ │ │  
 │ │ • React UI │ │ • Authentication│ │  
 │ │ • MSAL Auth │ │ • API Gateway │ │  
 │ │ • TypeScript │ │ • Request Proxy │ │  
 │ └─────────────────┘ └─────────┬───────┘ │  
 │ │ │  
 └────────────────────────────────────────┼─────────────────────────┘  
 │  
 ┌────────────────────────────────────────┼─────────────────────────┐  
 │ External Services │ │  
 │ ▼ │  
 │ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │  
 │ │ Azure AD │ │ Lab45 AI │ │ App Insights│ │  
 │ │ • Identity │ │ • GPT-3.5 │ │ • Monitoring│ │  
 │ │ • OAuth 2.0 │ │ • Streaming │ │ • Analytics │ │  
 │ └─────────────┘ └─────────────┘ └─────────────┘ │  
 └─────────────────────────────────────────────────────────────────┘

### **2.2 High-Level Architecture Layers**

┌─────────────────────────────────────────────────────────────────┐  
│ PRESENTATION LAYER │  
│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │  
│ │ AuthButton │ │ChatInterface│ │AuthContent │ │ UI Components│ │  
│ └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘ │  
└─────────────────────────────────────────────────────────────────┘  
│  
┌─────────────────────────────────────────────────────────────────┐  
│ APPLICATION LAYER │  
│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │  
│ │ Auth Hooks │ │ State Mgmt │ │ API Client │ │ Utils │ │  
│ └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘ │  
└─────────────────────────────────────────────────────────────────┘  
│  
┌─────────────────────────────────────────────────────────────────┐  
│ SERVICE LAYER │  
│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │  
│ │ API Routes │ │ Middleware │ │ Validation │ │ Error Handle│ │  
│ └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘ │  
└─────────────────────────────────────────────────────────────────┘  
│  
┌─────────────────────────────────────────────────────────────────┐  
│ INTEGRATION LAYER │  
│ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │  
│ │ Azure AD │ │ Lab45 AI │ │ Monitoring │ │ External │ │  
│ │ Integration │ │ Integration │ │ Integration │ │ Services │ │  
│ └─────────────┘ └─────────────┘ └─────────────┘ └─────────────┘ │  
└─────────────────────────────────────────────────────────────────┘

## **3. Functional Design**

### **3.1 Core Functional Components**

#### **3.1.1 Authentication Module**

// Authentication Module Design  
interface AuthenticationModule {  
 components: {  
 AuthButton: {  
 responsibility: "User login/logout UI";  
 dependencies: ["@azure/msal-react"];  
 state: ["isAuthenticated", "userAccount"];  
 actions: ["login", "logout"];  
 };  
   
 MsalProvider: {  
 responsibility: "Authentication context provider";  
 dependencies: ["@azure/msal-browser"];  
 configuration: "msal-config.ts";  
 scope: "Application-wide";  
 };  
   
 useAccessToken: {  
 responsibility: "Token management hook";  
 features: ["silent renewal", "token caching"];  
 errorHandling: "Fallback to interactive login";  
 };  
 };  
   
 flows: {  
 login: "Popup/Redirect → Azure AD → Token → App State";  
 logout: "Clear tokens → Redirect → Unauthenticated state";  
 silentRenewal: "Background token refresh";  
 };  
}

#### **3.1.2 Chat Interface Module**

// Chat Interface Module Design  
interface ChatInterfaceModule {  
 components: {  
 ChatInterface: {  
 responsibility: "Main chat UI and logic";  
 state: {  
 messages: "Array<{role: 'user'|'assistant', content: string}>";  
 input: "string";  
 isLoading: "boolean";  
 };  
 actions: ["sendMessage", "clearChat", "exportChat"];  
 };  
   
 MessageBubble: {  
 responsibility: "Individual message display";  
 variants: ["user", "assistant", "system"];  
 features: ["markdown support", "copy functionality"];  
 };  
   
 InputArea: {  
 responsibility: "Message input and submission";  
 features: ["auto-resize", "keyboard shortcuts", "file upload"];  
 validation: "Input sanitization and length limits";  
 };  
 };  
   
 dataFlow: {  
 userInput: "Input → Validation → API Call → Stream Response → UI Update";  
 streaming: "Server-sent events for real-time responses";  
 errorHandling: "Graceful error display and retry options";  
 };  
}  
 **3.1.3 API Service Module**

// API Service Module Design  
interface APIServiceModule {  
 routes: {  
 completions: {  
 path: "/api/completions";  
 method: "POST";  
 authentication: "Bearer token required";  
 requestBody: {  
 prompt: "string";  
 context?: "string";  
 parameters?: "AIParameters";  
 };  
 response: "Streaming JSON chunks";  
 };  
 };  
   
 middleware: {  
 authentication: "Token validation and user identification";  
 rateLimit: "Request throttling per user";  
 logging: "Request/response logging for monitoring";  
 errorHandling: "Centralized error processing";  
 };  
   
 integrations: {  
 lab45AI: "Proxy requests with authentication passthrough";  
 azureAD: "Token validation and user profile";  
 monitoring: "Telemetry and metrics collection";  
 };  
}

### **3.2 Component Interaction Design**

sequenceDiagram  
 participant U as User  
 participant AB as AuthButton  
 participant MP as MsalProvider  
 participant CI as ChatInterface  
 participant API as API Route  
 participant AI as Lab45 AI  
   
 U->>AB: Click Login  
 AB->>MP: Trigger Login  
 MP->>MP: Redirect to Azure AD  
 MP->>AB: Authentication Success  
 AB->>CI: Enable Chat Interface  
   
 U->>CI: Type Message  
 CI->>API: POST /api/completions  
 API->>API: Validate Token  
 API->>AI: Forward Request  
 AI-->>API: Stream Response  
 API-->>CI: Forward Stream  
 CI->>U: Display Response

**4. Technical Design**

### **4.1 Frontend Technical Architecture**

#### **4.1.1 Component Hierarchy**

App (layout.tsx)  
├── MsalProvider  
│ ├── AuthButton  
│ └── AuthContent  
│ └── ChatInterface  
│ ├── MessageList  
│ │ └── MessageBubble[]  
│ ├── InputArea  
│ │ ├── TextInput  
│ │ └── SendButton  
│ └── StatusIndicator

#### **4.1.2 State Management Design**

// State Management Architecture  
interface StateManagement {  
 global: {  
 authentication: {  
 isAuthenticated: boolean;  
 user: UserProfile | null;  
 accessToken: string | null;  
 tokenExpiry: Date | null;  
 };  
 };  
   
 local: {  
 chatInterface: {  
 messages: Message[];  
 currentInput: string;  
 isLoading: boolean;  
 error: string | null;  
 };  
 };  
   
 derived: {  
 canSendMessage: "isAuthenticated && !isLoading && input.length > 0";  
 shouldShowLogin: "!isAuthenticated";  
 tokenNeedsRefresh: "tokenExpiry < now + 5minutes";  
 };  
}

d timeouts";  
 };

**4.1.3 Hook Design Patterns**

// Custom Hooks Design  
interface CustomHooks {  
 useAccessToken: {  
 purpose: "Manage authentication tokens";  
 returns: "string | null";  
 sideEffects: ["token refresh", "error handling"];  
 dependencies: ["@azure/msal-react"];  
 };  
   
 useChat: {  
 purpose: "Manage chat state and API calls";  
 returns: {  
 messages: "Message[]";  
 sendMessage: "(message: string) => Promise<void>";  
 isLoading: "boolean";  
 error: "string | null";  
 };  
 optimizations: ["debounced input", "request cancellation"];  
 };  
   
 useStreaming: {  
 purpose: "Handle streaming responses";  
 returns: {  
 stream: "ReadableStream | null";  
 content: "string";  
 isComplete: "boolean";  
 };  
 errorHandling: "Connection failures an}

### **4.2 Backend Technical Architecture**

#### **4.2.1 API Route Design**

// API Route Implementation Pattern  
interface APIRouteDesign {  
 structure: {  
 handler: "Default export function";  
 methods: ["GET", "POST", "PUT", "DELETE"];  
 middleware: "Inline functions";  
 validation: "Runtime type checking";  
 };  
   
 implementation: {  
 authentication: {  
 pattern: "Bearer token in Authorization header";  
 validation: "MSAL token verification";  
 userContext: "Extract user information from token";  
 };  
   
 errorHandling: {  
 pattern: "Try-catch with typed errors";  
 responses: "Consistent error format";  
 logging: "Structured error logging";  
 };  
   
 streaming: {  
 pattern: "ReadableStream response";  
 chunking: "Server-sent events format";  
 errorRecovery: "Graceful stream termination";  
 };  
 };  
}

#### **4.2.2 Middleware Design**

// Middleware Architecture  
interface MiddlewareDesign {  
 authentication: {  
 order: 1;  
 responsibility: "Validate and extract user from token";  
 implementation: "MSAL token verification";  
 errorResponse: "401 Unauthorized";  
 };  
   
 rateLimit: {  
 order: 2;  
 responsibility: "Enforce request limits per user";  
 implementation: "In-memory sliding window";  
 errorResponse: "429 Too Many Requests";  
 };  
   
 validation: {  
 order: 3;  
 responsibility: "Validate request payload";  
 implementation: "TypeScript runtime validation";  
 errorResponse: "400 Bad Request";  
 };  
   
 logging: {  
 order: 4;  
 responsibility: "Log request/response for monitoring";  
 implementation: "Application Insights integration";  
 data: ["user\_id", "request\_path", "response\_time"];  
 };  
}

## **5. Database Design**

### **5.1 Current State (Stateless)**

// Current Stateless Design  
interface StatelessDesign {  
 storage: "None - fully stateless";  
 sessionData: "Client-side only (session storage)";  
 authentication: "Token-based, no server-side sessions";  
 conversations: "Not persisted (MVP limitation)";  
   
 implications: {  
 scalability: "Excellent - no database bottlenecks";  
 consistency: "Limited - no conversation history";  
 recovery: "Poor - data lost on browser refresh";  
 performance: "Excellent - no database queries";  
 };  
}

### **5.2 Future Database Design (Phase 2)**

// Future Database Schema Design  
interface DatabaseDesign {  
 tables: {  
 users: {  
 id: "UUID PRIMARY KEY";  
 azure\_id: "VARCHAR(255) UNIQUE";  
 email: "VARCHAR(255)";  
 display\_name: "VARCHAR(255)";  
 created\_at: "TIMESTAMP";  
 last\_login: "TIMESTAMP";  
 preferences: "JSON";  
 };  
   
 conversations: {  
 id: "UUID PRIMARY KEY";  
 user\_id: "UUID FOREIGN KEY";  
 title: "VARCHAR(255)";  
 created\_at: "TIMESTAMP";  
 updated\_at: "TIMESTAMP";  
 metadata: "JSON";  
 };  
   
 messages: {  
 id: "UUID PRIMARY KEY";  
 conversation\_id: "UUID FOREIGN KEY";  
 role: "ENUM('user', 'assistant', 'system')";  
 content: "TEXT";  
 timestamp: "TIMESTAMP";  
 metadata: "JSON";  
 };  
   
 user\_sessions: {  
 id: "UUID PRIMARY KEY";  
 user\_id: "UUID FOREIGN KEY";  
 session\_token: "VARCHAR(255)";  
 expires\_at: "TIMESTAMP";  
 created\_at: "TIMESTAMP";  
 };  
 };  
   
 indexes: {  
 users: ["azure\_id", "email"];  
 conversations: ["user\_id", "created\_at"];  
 messages: ["conversation\_id", "timestamp"];  
 user\_sessions: ["session\_token", "user\_id"];  
 };  
}

## **6. API Design**

### **6.1 RESTful API Design**

#### **6.1.1 Current API Endpoints**

// Current API Specification  
interface CurrentAPISpec {  
 "/api/completions": {  
 method: "POST";  
 authentication: "Bearer token required";  
 rateLimit: "60 requests/minute";  
   
 request: {  
 headers: {  
 "Authorization": "Bearer <access\_token>";  
 "Content-Type": "application/json";  
 };  
 body: {  
 prompt: string;  
 context?: string;  
 };  
 };  
   
 response: {  
 type: "Server-Sent Events";  
 format: "JSON chunks";  
 structure: {  
 data: {  
 content: string;  
 };  
 };  
 };  
   
 errors: {  
 400: "Bad Request - Invalid prompt";  
 401: "Unauthorized - Invalid or missing token";  
 429: "Too Many Requests - Rate limit exceeded";  
 503: "Service Unavailable - AI service error";  
 };  
 };  
}

#### **6.1.2 Future API Endpoints (Phase 2)**

// Future API Specification  
interface FutureAPISpec {  
 "/api/conversations": {  
 GET: {  
 purpose: "List user conversations";  
 response: "Array<ConversationSummary>";  
 pagination: "Cursor-based";  
 };  
 POST: {  
 purpose: "Create new conversation";  
 request: { title?: string };  
 response: "Conversation";  
 };  
 };  
   
 "/api/conversations/{id}": {  
 GET: {  
 purpose: "Get conversation details";  
 response: "ConversationWithMessages";  
 };  
 PUT: {  
 purpose: "Update conversation metadata";  
 request: "Partial<Conversation>";  
 };  
 DELETE: {  
 purpose: "Delete conversation";  
 response: "Success confirmation";  
 };  
 };  
   
 "/api/conversations/{id}/messages": {  
 GET: {  
 purpose: "Get conversation messages";  
 response: "Array<Message>";  
 pagination: "Offset-based";  
 };  
 POST: {  
 purpose: "Add message to conversation";  
 request: "MessageInput";  
 response: "Message";  
 };  
 };  
}

### **6.2 API Security Design**

// API Security Specification  
interface APISecurityDesign {  
 authentication: {  
 method: "OAuth 2.0 Bearer tokens";  
 provider: "Azure Active Directory";  
 validation: "MSAL token verification";  
 caching: "No server-side token storage";  
 };  
   
 authorization: {  
 model: "Resource-based access control";  
 rules: {  
 conversations: "User can only access own conversations";  
 messages: "User can only access messages in own conversations";  
 admin: "Special endpoints for admin users only";  
 };  
 };  
   
 rateLimiting: {  
 strategy: "Sliding window per user";  
 limits: {  
 completions: "60/minute";  
 conversations: "100/minute";  
 messages: "200/minute";  
 };  
 enforcement: "Return 429 with Retry-After header";  
 };  
   
 inputValidation: {  
 sanitization: "HTML encoding and XSS prevention";  
 lengthLimits: {  
 prompt: "10,000 characters";  
 title: "255 characters";  
 };  
 contentFiltering: "Basic profanity and spam detection";  
 };  
}

## **7. User Interface Design**

**7.1 UI Component Design**

**7.1.1 Component Specifications**

// UI Component Design Specifications  
interface UIComponentDesign {  
 AuthButton: {  
 states: ["loading", "authenticated", "unauthenticated", "error"];  
 variants: ["primary", "secondary"];  
 responsive: "Adaptive text and button size";  
 accessibility: "ARIA labels and keyboard navigation";  
   
 implementation: {  
 framework: "React functional component";  
 styling: "Tailwind CSS classes";  
 icons: "Built-in SVG icons";  
 animations: "CSS transitions";  
 };  
 };  
   
 ChatInterface: {  
 layout: "Flex column with fixed input area";  
 scrolling: "Auto-scroll to latest message";  
 responsive: "Mobile-first design";  
 accessibility: "Screen reader support";  
   
 subComponents: {  
 MessageList: "Virtual scrolling for performance";  
 MessageBubble: "Differentiated styling by role";  
 InputArea: "Auto-expanding textarea";  
 StatusIndicator: "Loading and error states";  
 };  
 };  
   
 MessageBubble: {  
 variants: ["user", "assistant", "system"];  
 features: ["timestamp", "copy button", "markdown rendering"];  
 styling: "Distinct colors and positioning";  
 animations: "Slide-in transition";  
 };  
}

#### **7.1.2 Responsive Design**

/\* Responsive Design Breakpoints \*/  
.responsive-design {  
 /\* Mobile First Approach \*/  
 --mobile: "320px - 767px";  
 --tablet: "768px - 1023px";  
 --desktop: "1024px+";  
   
 /\* Component Adaptations \*/  
 .chat-interface {  
 mobile: "Full screen, minimal padding";  
 tablet: "Centered with side margins";  
 desktop: "Max width with sidebar space";  
 }  
   
 .message-bubble {  
 mobile: "80% width, small font";  
 tablet: "70% width, medium font";  
 desktop: "60% width, larger font";  
 }  
   
 .input-area {  
 mobile: "Fixed bottom, full width";  
 tablet: "Fixed bottom, contained";  
 desktop: "Relative position";  
 }  
}  
 **7.2 User Experience Design**

**7.2.1 Interaction Flow**

stateDiagram-v2  
 [\*] --> Landing  
 Landing --> Authenticating: Click Login  
 Authenticating --> Authenticated: Success  
 Authenticating --> Error: Failure  
 Error --> Landing: Retry  
   
 Authenticated --> Chatting: Auto-navigate  
 Chatting --> Typing: User Input  
 Typing --> Sending: Submit Message  
 Sending --> Receiving: API Call  
 Receiving --> Chatting: Response Complete  
   
 Chatting --> Authenticated: Sign Out  
 Authenticated --> [\*]: Session End

#### **7.2.2 Error Handling UX**

// Error Handling UX Design  
interface ErrorHandlingUX {  
 categories: {  
 authentication: {  
 display: "Login button with error message";  
 action: "Redirect to Azure AD login";  
 recovery: "Automatic retry after login";  
 };  
   
 network: {  
 display: "Toast notification with retry button";  
 action: "Show cached content if available";  
 recovery: "Automatic retry with exponential backoff";  
 };  
   
 validation: {  
 display: "Inline error below input field";  
 action: "Highlight invalid input";  
 recovery: "Real-time validation feedback";  
 };  
   
 service: {  
 display: "Banner notification about service issues";  
 action: "Provide alternative actions";  
 recovery: "Automatic retry when service recovers";  
 };  
 };  
   
 principles: {  
 clarity: "Clear, jargon-free error messages";  
 actionable: "Always provide next steps";  
 contextual: "Show errors near relevant UI elements";  
 persistent: "Errors remain until resolved";  
 };  
}

**8. Security Design**

### **8.1 Authentication Security Design**

#### **8.1.1 MSAL Integration Security**

// MSAL Security Configuration  
interface MSALSecurityDesign {  
 configuration: {  
 clientId: "Application ID from Azure AD app registration";  
 authority: "[https://login.microsoftonline.com/{tenant-id](https://login.microsoftonline.com/%7btenant-id)}";  
 redirectUri: "<https://app-domain.com/>";  
 postLogoutRedirectUri: "<https://app-domain.com/>";  
   
 scopes: [  
 "openid", // Basic user identity  
 "profile", // User profile information  
 "User.Read", // Microsoft Graph user read  
 "api://app-id/.default" // Custom app permissions  
 ];  
 };  
   
 tokenSecurity: {  
 storage: "sessionStorage (secure, temporary)";  
 encryption: "Browser-native security";  
 expiration: "1 hour access tokens";  
 refresh: "Silent renewal in background";  
   
 validation: {  
 signature: "RSA256 signature verification";  
 issuer: "Azure AD tenant validation";  
 audience: "Application ID validation";  
 expiration: "Timestamp validation";  
 nonce: "CSRF protection";  
 };  
 };  
   
 securityHeaders: {  
 "Strict-Transport-Security": "max-age=31536000; includeSubDomains";  
 "X-Content-Type-Options": "nosniff";  
 "X-Frame-Options": "DENY";  
 "X-XSS-Protection": "1; mode=block";  
 "Content-Security-Policy": "default-src 'self'; connect-src 'self' <https://login.microsoftonline.com> <https://api.lab45.ai>";  
 };  
}

#### **8.1.2 API Security Design**

// API Security Implementation  
interface APISecurityDesign {  
 tokenValidation: {  
 process: [  
 "Extract Bearer token from Authorization header",  
 "Verify token signature using Azure AD public keys",  
 "Validate token claims (iss, aud, exp, nbf)",  
 "Extract user information from token claims",  
 "Check token permissions and scopes"  
 ];  
   
 caching: {  
 publicKeys: "Cache Azure AD public keys for 24 hours";  
 userInfo: "Cache user profile for request duration only";  
 };  
   
 errorHandling: {  
 invalidToken: "Return 401 with WWW-Authenticate header";  
 expiredToken: "Return 401 with token refresh guidance";  
 insufficientScope: "Return 403 with required scopes";  
 };  
 };  
   
 requestSecurity: {  
 rateLimiting: {  
 algorithm: "Sliding window";  
 storage: "In-memory (Redis for scaling)";  
 limits: "Per-user and per-endpoint";  
 response: "429 with Retry-After header";  
 };  
   
 inputValidation: {  
 sanitization: "HTML encoding, script removal";  
 lengthLimits: "Enforce maximum input sizes";  
 typeValidation: "Runtime TypeScript validation";  
 contentFiltering: "Basic inappropriate content detection";  
 };  
 };  
}

### **8.2 Data Protection Design**

// Data Protection Implementation  
interface DataProtectionDesign {  
 encryption: {  
 inTransit: {  
 protocol: "TLS 1.3 minimum";  
 certificates: "Let's Encrypt with auto-renewal";  
 ciphers: "AEAD ciphers only (AES-GCM, ChaCha20-Poly1305)";  
 hsts: "Strict Transport Security enforced";  
 };  
   
 atRest: {  
 tokens: "Browser sessionStorage (encrypted by browser)";  
 cache: "No persistent storage in current version";  
 logs: "Application Insights encryption";  
 backups: "Not applicable (stateless design)";  
 };  
 };  
   
 privacy: {  
 dataMinimization: "Only collect necessary user data";  
 retention: "No persistent conversation storage";  
 anonymization: "Remove PII from logs and analytics";  
 consent: "Clear privacy policy and consent flow";  
   
 gdprCompliance: {  
 rightToAccess: "User can view their data via Azure AD";  
 rightToErasure: "No persistent data to erase";  
 rightToPortability: "Export conversation feature";  
 dataProcessingLawfulness: "Legitimate interest and consent";  
 };  
 };  
}

## **9. Performance Design**

### **9.1 Frontend Performance Design**

#### **9.1.1 Optimization Strategies**

// Frontend Performance Optimization  
interface FrontendPerformanceDesign {  
 bundleOptimization: {  
 codesplitting: {  
 routes: "Automatic page-based splitting";  
 components: "Dynamic imports for heavy components";  
 libraries: "Separate vendor bundle";  
 };  
   
 treeshaking: "Remove unused code";  
 minification: "Terser for JavaScript, cssnano for CSS";  
 compression: "Gzip and Brotli compression";  
   
 bundleAnalysis: {  
 tool: "webpack-bundle-analyzer";  
 targets: {  
 main: "< 200KB gzipped";  
 vendor: "< 300KB gzipped";  
 total: "< 500KB gzipped";  
 };  
 };  
 };  
   
 runtimeOptimization: {  
 reactOptimization: {  
 memoization: "React.memo for expensive components";  
 callbacks: "useCallback for stable function references";  
 effects: "useEffect dependency optimization";  
 virtualization: "Virtual scrolling for message lists";  
 };  
   
 imageOptimization: {  
 format: "WebP with fallbacks";  
 sizing: "Responsive images with srcset";  
 loading: "Lazy loading with intersection observer";  
 placeholder: "Low-quality image placeholders";  
 };  
   
 caching: {  
 strategy: "Cache-first for static assets";  
 serviceWorker: "Workbox for offline capabilities";  
 apiCaching: "Short-term response caching";  
 browserCache: "Leverage browser caching directives";  
 };  
 };  
}

#### **9.1.2 Loading Performance**

// Loading Performance Strategy  
interface LoadingPerformanceDesign {  
 initialLoad: {  
 criticalPath: [  
 "HTML parsing",  
 "CSS loading and parsing",  
 "JavaScript loading and execution",  
 "React hydration",  
 "Authentication check"  
 ];  
   
 optimization: {  
 preload: "Critical CSS and JavaScript";  
 prefetch: "Next page resources";  
 inlining: "Critical CSS inline";  
 defer: "Non-critical JavaScript deferred";  
 };  
   
 metrics: {  
 fcp: "< 1.5 seconds (First Contentful Paint)";  
 lcp: "< 2.5 seconds (Largest Contentful Paint)";  
 fid: "< 100ms (First Input Delay)";  
 cls: "< 0.1 (Cumulative Layout Shift)";  
 };  
 };  
   
 streamingResponse: {  
 implementation: "Server-Sent Events";  
 chunking: "Progressive message building";  
 buffering: "Minimal buffering for responsiveness";  
 errorRecovery: "Graceful stream interruption handling";  
   
 performance: {  
 latency: "< 100ms first chunk";  
 throughput: "Process chunks as received";  
 memory: "Bounded memory usage";  
 };  
 };  
}

### **9.2 Backend Performance Design**

// Backend Performance Strategy  
interface BackendPerformanceDesign {  
 apiPerformance: {  
 responseTime: {  
 target: "< 200ms for auth validation";  
 streaming: "< 1 second to first AI response chunk";  
 monitoring: "P95 response time tracking";  
 };  
   
 concurrency: {  
 architecture: "Event-driven, non-blocking I/O";  
 limits: "1000 concurrent connections per instance";  
 queuing: "Request queuing for overload protection";  
 };  
   
 caching: {  
 strategy: "Cache-aside pattern";  
 targets: {  
 userProfiles: "1 hour TTL";  
 authTokens: "No caching (security)";  
 staticContent: "Indefinite with versioning";  
 };  
 };  
 };  
   
 externalIntegration: {  
 aiService: {  
 timeout: "30 seconds maximum";  
 retries: "3 attempts with exponential backoff";  
 circuitBreaker: "Fail fast after consecutive failures";  
 fallback: "Graceful degradation message";  
 };  
   
 azureAD: {  
 timeout: "10 seconds for token validation";  
 caching: "Public key caching for 24 hours";  
 retries: "2 attempts for network errors";  
 };  
 };  
}

## **10. Integration Design**

### **10.1 External Service Integration**

#### **10.1.1 Lab45 AI Integration**

// Lab45 AI Integration Design  
interface Lab45AIIntegration {  
 configuration: {  
 baseURL: "<https://api.lab45.ai/v1.1>";  
 endpoint: "/skills/completion/query";  
 timeout: 30000; // 30 seconds  
 retries: 3;  
 backoffMultiplier: 2;  
 };  
   
 requestFormat: {  
 method: "POST";  
 headers: {  
 "Authorization": "Bearer <passthrough\_token>";  
 "Content-Type": "application/json";  
 };  
 body: {  
 messages: Array<{  
 role: "user" | "assistant" | "system";  
 content: string;  
 }>;  
 skill\_parameters: {  
 model\_name: "gpt-35-turbo-16k";  
 max\_output\_tokens: 1000;  
 temperature: 0.7;  
 stream: true;  
 };  
 };  
 };  
   
 responseHandling: {  
 streaming: {  
 format: "Server-Sent Events";  
 parsing: "JSON chunks separated by newlines";  
 accumulation: "Progressive content building";  
 completion: "End-of-stream detection";  
 };  
   
 errorHandling: {  
 networkErrors: "Retry with exponential backoff";  
 authErrors: "Return 401 to client";  
 rateLimitErrors: "Return 429 to client";  
 serviceErrors: "Return 503 with retry information";  
 };  
 };  
   
 security: {  
 tokenPassthrough: "Forward user's Azure AD token";  
 noTokenStorage: "Never store tokens server-side";  
 requestValidation: "Validate request format before forwarding";  
 responseFiltering: "Basic content filtering";  
 };  
}

#### **10.1.2 Azure AD Integration**

// Azure AD Integration Design  
interface AzureADIntegration {  
 configuration: {  
 tenantId: "Organization's Azure AD tenant ID";  
 clientId: "Application ID from app registration";  
 clientSecret: "Not used (public client)";  
 scopes: ["openid", "profile", "User.Read"];  
 };  
   
 authenticationFlow: {  
 frontend: {  
 library: "@azure/msal-react";  
 flow: "Authorization Code with PKCE";  
 storage: "sessionStorage for tokens";  
 silentRenewal: "Background token refresh";  
 };  
   
 backend: {  
 validation: {  
 library: "@azure/msal-node";  
 publicKeys: "Fetch from Azure AD JWKS endpoint";  
 caching: "Cache public keys for 24 hours";  
 claims: "Validate standard and custom claims";  
 };  
 };  
 };  
   
 userProfile: {  
 source: "ID token claims";  
 fields: ["sub", "name", "email", "preferred\_username"];  
 caching: "Request-scoped only";  
 privacy: "No persistent storage";  
 };  
   
 errorHandling: {  
 loginFailure: "Redirect to error page with retry option";  
 tokenExpired: "Automatic silent renewal attempt";  
 networkError: "Retry with exponential backoff";  
 serviceError: "Fallback to cached tokens if available";  
 };  
}

### **10.2 Monitoring Integration**

// Monitoring Integration Design  
interface MonitoringIntegration {  
 applicationInsights: {  
 configuration: {  
 connectionString: "Environment-specific connection string";  
 sampling: 100; // 100% sampling for MVP  
 telemetryProcessors: ["Filtering PII data"];  
 };  
   
 telemetryTypes: {  
 requests: "All HTTP requests with response times";  
 dependencies: "External service calls";  
 exceptions: "All unhandled exceptions";  
 customEvents: "Business logic events";  
 pageViews: "Frontend page navigation";  
 performance: "Core Web Vitals";  
 };  
   
 customMetrics: {  
 authentication: {  
 loginAttempts: "Counter";  
 loginSuccessRate: "Percentage";  
 tokenRefreshRate: "Counter";  
 };  
   
 chat: {  
 messagesPerSession: "Distribution";  
 responseTime: "Duration";  
 errorRate: "Percentage";  
 userSatisfaction: "Rating";  
 };  
   
 performance: {  
 apiResponseTime: "Duration";  
 streamingLatency: "Duration";  
 concurrentUsers: "Gauge";  
 };  
 };  
 };  
   
 alerting: {  
 rules: {  
 errorRate: "Alert if error rate > 1% over 5 minutes";  
 responseTime: "Alert if P95 response time > 5 seconds";  
 availability: "Alert if availability < 99.9%";  
 authFailures: "Alert if auth failure rate > 10%";  
 };  
   
 channels: {  
 email: "Development team distribution list";  
 slack: "Alert channel for immediate notifications";  
 sms: "Critical alerts for on-call personnel";  
 };  
 };  
}  
 **11. Error Handling Design**

**11.1 Error Classification and Handling**

// Error Handling Framework  
interface ErrorHandlingDesign {  
 errorTypes: {  
 AuthenticationError: {  
 scenarios: ["Invalid token", "Expired token", "Missing token"];  
 handling: "Redirect to login";  
 userMessage: "Please sign in to continue";  
 httpStatus: 401;  
 retryable: true;  
 };  
   
 AuthorizationError: {  
 scenarios: ["Insufficient permissions", "Blocked user"];  
 handling: "Show access denied message";  
 userMessage: "You don't have permission to access this feature";  
 httpStatus: 403;  
 retryable: false;  
 };  
   
 ValidationError: {  
 scenarios: ["Invalid input", "Missing required fields"];  
 handling: "Show inline validation errors";  
 userMessage: "Please check your input and try again";  
 httpStatus: 400;  
 retryable: true;  
 };  
   
 NetworkError: {  
 scenarios: ["Connection timeout", "DNS resolution failure"];  
 handling: "Retry with exponential backoff";  
 userMessage: "Connection problem. Retrying...";  
 httpStatus: 0;  
 retryable: true;  
 };  
   
 ServiceError: {  
 scenarios: ["External service unavailable", "Rate limit exceeded"];  
 handling: "Show service unavailable message";  
 userMessage: "Service temporarily unavailable";  
 httpStatus: 503;  
 retryable: true;  
 };  
 };  
   
 errorBoundaries: {  
 application: {  
 scope: "Entire application";  
 fallback: "Generic error page with refresh option";  
 logging: "Log all unhandled React errors";  
 };  
   
 chatInterface: {  
 scope: "Chat component tree";  
 fallback: "Chat unavailable message with retry";  
 recovery: "Reset chat state and retry";  
 };  
   
 authentication: {  
 scope: "Authentication components";  
 fallback: "Login prompt";  
 recovery: "Clear auth state and redirect to login";  
 };  
 };  
}

### **11.2 Error Recovery Strategies**

graph TD  
 A[Error Detected] --> B{Error Type}  
   
 B --> C[Authentication Error]  
 B --> D[Network Error]  
 B --> E[Service Error]  
 B --> F[Validation Error]  
   
 C --> G[Clear Auth State]  
 G --> H[Redirect to Login]  
 H --> I[User Re-authenticates]  
 I --> J[Retry Original Action]  
   
 D --> K[Exponential Backoff]  
 K --> L{Max Retries?}  
 L -->|No| M[Wait and Retry]  
 L -->|Yes| N[Show Error Message]  
 M --> A  
   
 E --> O[Check Service Status]  
 O --> P{Service Available?}  
 P -->|Yes| M  
 P -->|No| Q[Degraded Mode]  
   
 F --> R[Show Validation Error]  
 R --> S[User Corrects Input]  
 S --> T[Revalidate]  
 T --> J  
   
 J --> U[Success]  
 N --> V[Manual Retry Option]  
 Q --> V  
 V --> A

## **12. Deployment Design**

### **12.1 Environment Configuration**

// Environment Configuration Design  
interface DeploymentDesign {  
 environments: {  
 development: {  
 platform: "Local development server";  
 url: "<http://localhost:3000>";  
 authentication: {  
 tenantId: "development-tenant-id";  
 clientId: "development-client-id";  
 redirectUri: "<http://localhost:3000>";  
 };  
 aiService: "Development endpoint";  
 monitoring: "Console logging only";  
 ssl: false;  
 };  
   
 staging: {  
 platform: "Vercel preview deployment";  
 url: "<https://chatapp-lab45-staging.vercel.app>";  
 authentication: {  
 tenantId: "staging-tenant-id";  
 clientId: "staging-client-id";  
 redirectUri: "<https://chatapp-lab45-staging.vercel.app>";  
 };  
 aiService: "Staging endpoint";  
 monitoring: "Application Insights (staging)";  
 ssl: true;  
 };  
   
 production: {  
 platform: "Vercel production deployment";  
 url: "<https://chatapp-lab45.com>";  
 authentication: {  
 tenantId: "production-tenant-id";  
 clientId: "production-client-id";  
 redirectUri: "<https://chatapp-lab45.com>";  
 };  
 aiService: "Production endpoint";  
 monitoring: "Application Insights (production)";  
 ssl: true;  
 cdn: "Global edge network";  
 };  
 };  
   
 configuration: {  
 environmentVariables: {  
 AZURE\_AD\_CLIENT\_ID: "Azure AD application ID";  
 AZURE\_AD\_TENANT\_ID: "Azure AD tenant ID";  
 AZURE\_AD\_AUTHORITY: "Authority URL";  
 LAB45\_AI\_BASE\_URL: "AI service base URL";  
 APPLICATIONINSIGHTS\_CONNECTION\_STRING: "Monitoring connection";  
 ENVIRONMENT: "Environment identifier";  
 };  
   
 buildConfiguration: {  
 nextjs: {  
 typescript: true;  
 eslint: true;  
 tailwindcss: true;  
 bundleAnalyzer: "Development only";  
 };  
   
 optimization: {  
 minification: "Production only";  
 sourceMap: "Development and staging";  
 compression: "Production only";  
 treeshaking: "All environments";  
 };  
 };  
 };  
}

### **12.2 CI/CD Pipeline Design**

# CI/CD Pipeline Configuration  
name: Deploy ChatApp-LAB45  
on:  
 push: branches: [main, develop]  
 pull\_request:  
 branches: [main]  
  
jobs:  
 test:  
 runs-on: ubuntu-latest  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v3  
   
 - name: Setup Node.js  
 uses: actions/setup-node@v3  
 with:  
 node-version: '18'  
 cache: 'npm'  
   
 - name: Install dependencies  
 run: npm ci  
   
 - name: Type checking  
 run: npm run type-check  
   
 - name: Linting  
 run: npm run lint  
   
 - name: Unit tests  
 run: npm run test  
   
 - name: Security audit  
 run: npm audit --audit-level moderate  
   
 build:  
 needs: test  
 runs-on: ubuntu-latest  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v3  
   
 - name: Setup Node.js  
 uses: actions/setup-node@v3  
 with:  
 node-version: '18'  
 cache: 'npm'  
   
 - name: Install dependencies  
 run: npm ci  
   
 - name: Build application  
 run: npm run build  
 env:  
 AZURE\_AD\_CLIENT\_ID: ${{ secrets.AZURE\_AD\_CLIENT\_ID }}  
 AZURE\_AD\_TENANT\_ID: ${{ secrets.AZURE\_AD\_TENANT\_ID }}  
   
 - name: Bundle analysis  
 run: npm run analyze  
   
 deploy-staging:  
 needs: build  
 if: github.ref == 'refs/heads/develop'  
 runs-on: ubuntu-latest  
 steps:  
 - name: Deploy to Vercel Staging  
 uses: amondnet/vercel-action@v25  
 with:  
 vercel-token: ${{ secrets.VERCEL\_TOKEN }}  
 vercel-org-id: ${{ secrets.VERCEL\_ORG\_ID }}  
 vercel-project-id: ${{ secrets.VERCEL\_PROJECT\_ID }}  
   
 deploy-production:  
 needs: build  
 if: github.ref == 'refs/heads/main'  
 runs-on: ubuntu-latest  
 steps:  
 - name: Deploy to Vercel Production  
 uses: amondnet/vercel-action@v25  
 with:  
 vercel-token: ${{ secrets.VERCEL\_TOKEN }}  
 vercel-org-id: ${{ secrets.VERCEL\_ORG\_ID }}  
 vercel-project-id: ${{ secrets.VERCEL\_PROJECT\_ID }}  
 vercel-args: '--prod'  
   
 - name: Run E2E tests  
 run: npm run test:e2e  
 env:  
 TEST\_URL: <https://chatapp-lab45.com>

### **12.3 Infrastructure as Code**

// Infrastructure Configuration  
interface InfrastructureAsCode {  
vercelConfiguration: {  
 framework: "nextjs";  
 buildCommand: "npm run build";  
 outputDirectory: ".next";  
 installCommand: "npm ci";  
 devCommand: "npm run dev";  
   
 environmentVariables: {  
 production: {  
 AZURE\_AD\_CLIENT\_ID: "Production client ID";  
 AZURE\_AD\_TENANT\_ID: "Production tenant ID";  
 LAB45\_AI\_BASE\_URL: "Production AI service URL";  
 APPLICATIONINSIGHTS\_CONNECTION\_STRING: "Production monitoring";  
 };  
   
 preview: {  
 AZURE\_AD\_CLIENT\_ID: "Staging client ID";  
 AZURE\_AD\_TENANT\_ID: "Staging tenant ID";  
 LAB45\_AI\_BASE\_URL: "Staging AI service URL";  
 APPLICATIONINSIGHTS\_CONNECTION\_STRING: "Staging monitoring";  
 };  
 };  
   
 functions: {  
 "/api/\*": {  
 runtime: "nodejs18.x";  
 memory: 1024;  
 timeout: 30;  
 regions: ["all"];  
 };  
 };  
   
 headers: {  
 "/\*\*": {  
 "X-Frame-Options": "DENY";  
 "X-Content-Type-Options": "nosniff";  
 "Referrer-Policy": "strict-origin-when-cross-origin";  
 "Permissions-Policy": "camera=(), microphone=(), geolocation=()";  
 };  
 };  
   
 redirects: {  
 source: "/login";  
 destination: "/";  
 permanent: false;  
 };  
 };  
   
 monitoring: {  
 applicationInsights: {  
 workspace: "ChatApp-LAB45";  
 retentionInDays: 90;  
 samplingPercentage: 100;  
   
 alertRules: {  
 highErrorRate: "error\_rate > 1%";  
 slowResponse: "response\_time > 5s";  
 lowAvailability: "availability < 99.9%";  
 };  
 };  
 };  
}

## **Conclusion**

This High-Level Design document provides comprehensive technical specifications for the ChatApp-LAB45 enterprise AI chat application. The design emphasizes:

### **Design Principles Achieved**

1. **Modular Architecture**: Clear separation of concerns with well-defined component boundaries
2. **Security by Design**: Enterprise-grade authentication and authorization throughout
3. **Scalable Foundation**: Cloud-native architecture ready for horizontal scaling
4. **Performance Optimized**: Efficient frontend and backend performance strategies
5. **Maintainable Codebase**: TypeScript, clear interfaces, and comprehensive error handling

### **Technical Excellence**

* **Modern Stack**: Next.js, React, TypeScript for developer productivity
* **Enterprise Integration**: Seamless Azure AD and Microsoft ecosystem integration
* **Real-time Features**: Streaming AI responses for optimal user experience
* **Comprehensive Monitoring**: Full observability with Application Insights
* **Production Ready**: Complete CI/CD pipeline and deployment strategy

### **Future Extensibility**

The design provides a solid foundation for:

* Phase 2 features (conversation persistence, multi-model support)
* Enterprise enhancements (admin dashboard, advanced analytics)
* Scaling capabilities (microservices evolution, database integration)
* Security enhancements (advanced compliance, audit logging)

This HLD serves as the technical blueprint for development teams to implement the ChatApp-LAB45 system according to enterprise standards and best practices.