# **Development Artifacts Documentation**

## **Project Overview**

**Project Name:** CHATAPP-LAB45

**Type:** Next.js 15 Chat Application with Microsoft Azure Integration

**Framework:** React 18 with TypeScript

**Documentation Date:** July 1, 2025

This document provides a comprehensive overview of all development artifacts, tools, configurations, and processes used in the CHATAPP project.

## **Project Architecture**

### **Core Framework**

* **Next.js 15.1.0** - React-based full-stack framework
* **React 18** - Frontend UI library
* **TypeScript 5** - Type-safe JavaScript development
* **Tailwind CSS 3.4.1** - Utility-first CSS framework

### **Authentication & Integration**

* **Microsoft Azure AD** integration via MSAL (Microsoft Authentication Library)
  + @azure/msal-browser ^3.28.0
  + @azure/msal-react ^2.2.0
* **AI360 API** integration for chat functionality

### **Directory Structure**

CHATAPP-LAB45/  
├── config/ # Environment configurations  
├── public/ # Static assets  
├── scripts/ # Deployment & automation scripts  
├── .vscode/ # VS Code workspace settings  
├── .next/ # Next.js build output  
└── node\_modules/ # Dependencies

## **Development Tools & Dependencies**

### **Production Dependencies**

{  
 "@azure/msal-browser": "^3.28.0",  
 "@azure/msal-react": "^2.2.0",  
 "next": "15.1.0",  
 "react": "^18.0.0",  
 "react-dom": "^18.0.0"  
}

### **Development Dependencies**

#### **TypeScript & Type Definitions**

* @types/node ^20
* @types/react ^19
* @types/react-dom ^19
* typescript ^5

#### **Code Quality Tools**

* @typescript-eslint/eslint-plugin ^6.0.0
* @typescript-eslint/parser ^6.0.0
* eslint ^8.0.0
* eslint-config-next ^14.0.0
* prettier ^3.0.0

#### **Testing Framework**

* @testing-library/react ^13.0.0
* @testing-library/jest-dom ^5.0.0
* @testing-library/user-event ^14.0.0
* @playwright/test ^1.40.0
* jest ^29.0.0
* jest-environment-jsdom ^29.0.0

#### **Git Hooks & Automation**

* husky ^8.0.0 - Git hooks management
* lint-staged ^13.0.0 - Pre-commit linting

#### **Styling & Build Tools**

* postcss ^8
* tailwindcss ^3.4.1

## **Configuration Files**

### **1. TypeScript Configuration (tsconfig.json)**

{  
 "compilerOptions": {  
 "target": "ES2017",  
 "lib": ["dom", "dom.iterable", "esnext"],  
 "strict": true,  
 "noEmit": true,  
 "jsx": "preserve",  
 "incremental": true,  
 "moduleResolution": "bundler",  
 "paths": {  
 "@/\*": ["./\*"]  
 }  
 }  
}

**Key Features:**

* Strict TypeScript checking enabled
* Path aliases for clean imports (@/\*)
* Incremental compilation for faster builds
* Next.js plugin integration

### **2. ESLint Configuration (.eslintrc.json)**

**Extends:**

* next/core-web-vitals
* @typescript-eslint/recommended
* eslint:recommended

**Key Rules:**

* TypeScript strict rules (no unused vars, explicit typing)
* React Hooks linting
* Code style enforcement
* Import/export validation

### **3. Prettier Configuration (.prettierrc)**

* Code formatting standardization
* Integrated with ESLint for consistent styling

### **4. Tailwind CSS Configuration (tailwind.config.ts)**

* Utility-first CSS framework setup
* Custom design tokens and responsive breakpoints

### **5. PostCSS Configuration (postcss.config.mjs)**

* CSS processing pipeline
* Tailwind CSS integration

### **6. Next.js Configuration (next.config.ts)**

* Framework-specific build and runtime settings

## **Scripts & Automation**

### **Development Scripts**

npm run dev # Start development server  
npm run build # Build production bundle  
npm run start # Start production server  
npm run type-check # TypeScript type checking

### **Code Quality Scripts**

npm run lint # Run ESLint  
npm run lint:fix # Auto-fix ESLint issues  
npm run lint:strict # Strict linting (max 0 warnings)  
npm run format # Format code with Prettier  
npm run format:check # Check code formatting

### **Testing Scripts**

npm run test # Run all tests  
npm run test:watch # Run tests in watch mode  
npm run test:coverage # Generate test coverage report  
npm run test:unit # Run unit tests only  
npm run test:integration # Run integration tests only  
npm run test:e2e # Run Playwright E2E tests  
npm run test:e2e:ui # Run E2E tests with UI

### **Deployment Scripts**

npm run deploy:staging # Deploy to staging environment  
npm run deploy:production # Deploy to production environment

### **Git Hooks**

npm run pre-commit # Pre-commit validation

## **Code Quality & Standards**

### **Linting Strategy**

* **ESLint** with TypeScript integration
* **Next.js specific rules** for optimal React development
* **Strict mode** available for zero-warning builds

### **Code Formatting**

* **Prettier** for consistent code formatting
* **Automatic formatting** on save (VS Code integration)
* **Pre-commit formatting** via lint-staged

### **Pre-commit Hooks (Husky + lint-staged)**

{  
 "\*.{ts,tsx,js,jsx}": [  
 "eslint --fix",  
 "prettier --write"  
 ],  
 "\*.{css,scss,md,json}": [  
 "prettier --write"  
 ]  
}

### **TypeScript Standards**

* **Strict mode** enabled for type safety
* **Explicit function return types** optional
* **No unused variables** enforced
* **Consistent import styles** with path aliases

## **Testing Strategy**

### **Testing Pyramid**

#### **1. Unit Tests (Jest + React Testing Library)**

* **Framework:** Jest with jsdom environment
* **Tools:** React Testing Library for component testing
* **Location:** \_\_tests\_\_/unit/
* **Coverage:** Enabled with coverage reporting

#### **2. Integration Tests**

* **Framework:** Jest
* **Location:** \_\_tests\_\_/integration/
* **Focus:** API endpoints, component interactions

#### **3. End-to-End Tests (Playwright)**

* **Framework:** Playwright
* **Features:** Cross-browser testing
* **UI Mode:** Available for interactive debugging
* **Coverage:** User workflows and critical paths

### **Test Utilities**

* **@testing-library/user-event** for user interaction simulation
* **@testing-library/jest-dom** for custom Jest matchers

## **Deployment Pipeline**

### **Deployment Script (scripts/deploy.sh)**

**Features:**

* Multi-environment support (staging/production)
* Version management
* Colored logging and error handling
* Automated deployment process

**Environments:**

* **Staging:** chat-app-staging.vercel.app
* **Production:** chat-app-prod.vercel.app

### **Environment-Specific Configurations**

Each environment has dedicated settings for:

* API endpoints
* Debug modes
* Analytics tracking
* Azure AD integration
* AI360 API configuration

## **Development Workflow**

### **1. Local Development**

# Clone and setup  
git clone <repository>  
cd CHATAPP-LAB45  
npm install  
  
# Start development  
npm run dev

### **2. Code Quality Checks**

# Type checking  
npm run type-check  
  
# Linting  
npm run lint:strict  
  
# Formatting  
npm run format:check

### **3. Testing**

# Unit tests  
npm run test:unit  
  
# Integration tests  
npm run test:integration  
  
# E2E tests  
npm run test:e2e

### **4. Pre-commit Process**

* **Automatic:** Husky triggers lint-staged
* **Validation:** ESLint + Prettier on staged files
* **Type Check:** TypeScript compilation verification

### **5. Deployment**

# Staging deployment  
npm run deploy:staging  
  
# Production deployment  
npm run deploy:production

## **Environment Management**

### **Configuration Structure (config/environments.ts)**

#### **Development Environment**

* **API URL:** <http://localhost:3000/api>
* **Debug:** Enabled
* **Analytics:** Disabled
* **Azure AD:** Development tenant
* **AI360:** Development API endpoint

#### **Staging Environment**

* **API URL:** <https://chat-app-staging.vercel.app/api>
* **Debug:** Enabled
* **Analytics:** Disabled
* **Azure AD:** Staging tenant
* **AI360:** Production API endpoint

#### **Production Environment**

* **API URL:** <https://chat-app-prod.vercel.app/api>
* **Debug:** Disabled
* **Analytics:** Enabled
* **Azure AD:** Production tenant
* **AI360:** Production API endpoint

### **Environment Variables**

Required environment variables:

* AZURE\_CLIENT\_ID - Azure AD application client ID
* AZURE\_TENANT\_ID - Azure AD tenant ID
* AI360\_API\_URL - AI360 API endpoint (optional, defaults provided)

## **IDE Configuration**

### **VS Code Settings (.vscode/settings.json)**

#### **Code Quality Integration**

* **Format on Save:** Enabled
* **ESLint Auto-fix:** On save
* **Import Organization:** Automatic

#### **TypeScript Configuration**

* **Quote Style:** Double quotes
* **Auto Imports:** Enabled
* **Relative Import Paths:** Preferred

#### **File Exclusions**

* Build artifacts (.next/, out/, dist/)
* Dependencies (node\_modules/)
* Version control (.git/)

#### **Search Optimization**

* Excluded build directories from search
* Focused search on source code only

## **Build Artifacts**

### **Generated Files & Directories**

#### **Build Output**

* .next/ - Next.js build cache and output
* out/ - Static export output (if used)
* dist/ - Distribution build (if configured)

#### **Dependency Management**

* node\_modules/ - NPM dependencies
* package-lock.json - Dependency version lock

#### **Development Cache**

* .eslintcache - ESLint cache for faster linting
* TypeScript incremental compilation cache

## **Security & Best Practices**

### **Code Security**

* **TypeScript strict mode** prevents common runtime errors
* **ESLint security rules** catch potential vulnerabilities
* **Dependency auditing** via npm audit

### **Authentication Security**

* **Microsoft Azure AD** integration for enterprise security
* **MSAL library** for secure token management
* **Environment-specific** Azure configurations

### **Build Security**

* **Production builds** exclude debug information
* **Environment variables** for sensitive configuration
* **Staging environment** for pre-production testing

## **Maintenance & Updates**

### **Regular Maintenance Tasks**

1. **Dependency Updates:** Monthly review of package updates
2. **Security Audits:** Weekly npm audit checks
3. **Code Quality Reviews:** Continuous via automated tools
4. **Performance Monitoring:** Build time and bundle size tracking

### **Update Procedures**

1. **Major Updates:** Test in staging before production
2. **Security Updates:** Apply immediately after testing
3. **Framework Updates:** Follow Next.js upgrade guides
4. **TypeScript Updates:** Validate type definitions

## **Troubleshooting Guide**

### **Common Issues**

#### **Build Failures**

* Check TypeScript compilation: npm run type-check
* Verify ESLint rules: npm run lint
* Clear Next.js cache: rm -rf .next

#### **Test Failures**

* Update test snapshots if UI changed
* Check environment variables for integration tests
* Verify Playwright browser installation

#### **Deployment Issues**

* Validate environment configurations
* Check deployment script permissions
* Verify API endpoint accessibility