Project Plan Presentation (PPP)

ChatGPT Clone – Individual Project

📋 Executive Summary

A real-time conversational AI application that replicates ChatGPT functionality using React, Node.js Express, and the OpenAI API. The project demonstrates full-stack development with WebSocket communication and client-side session management.

🎯 Project Overview

Attribute	Details
Project Name	ChatGPT Clone
Type	Full-Stack Web Application
Duration	11 Weeks (2 Sprints)
Target Users	End users seeking AI-powered conversations

🛠️ Technology Stack

Frontend

- **React** – Component-based UI framework
- **IndexedDB** – Client-side persistent storage

Backend

- **Node.js Express** – Server framework
- **WebSocket** – Real-time bidirectional communication

External Services

- **OpenAI ChatGPT API** – AI response generation

✨ Key Features

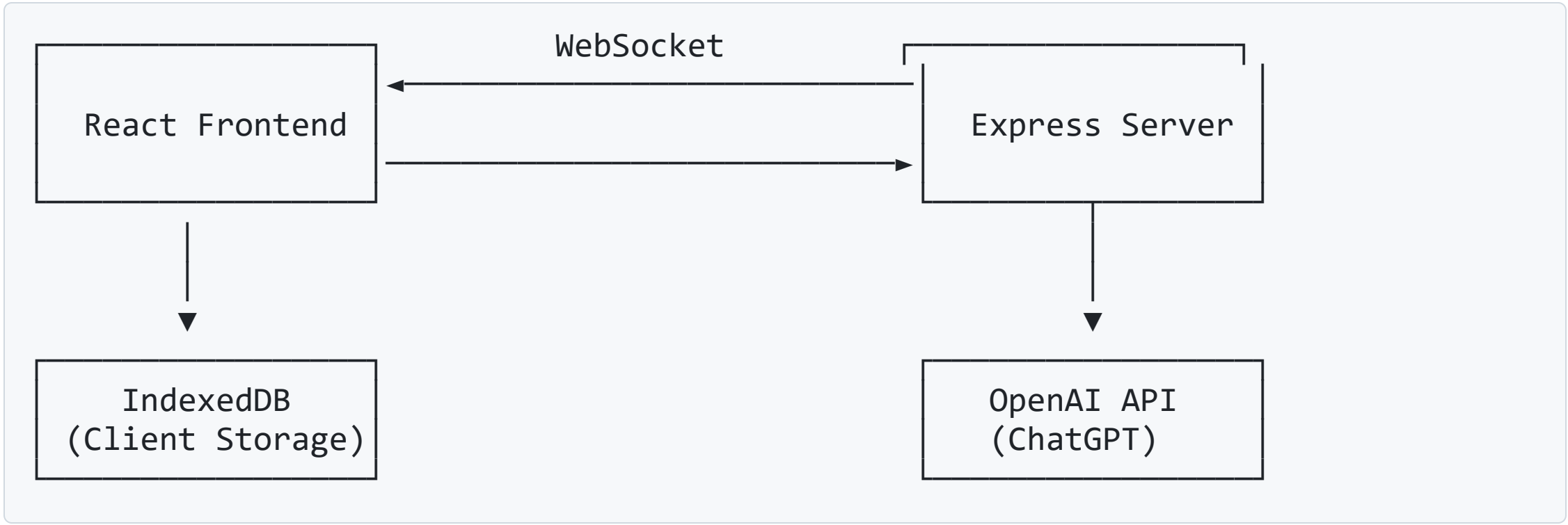
MVP Features (Sprint 1)

Feature	Description
Real-Time Messaging	WebSocket-based instant communication
Conversation History	Persistent storage of all chat sessions
Multi-Conversation Support	Create and switch between multiple chats
Session Management	Client-side sessions with IndexedDB
AI Integration	Direct OpenAI API integration for responses

Enhanced Features (Sprint 2)

Feature	Description
Dark/Light Mode	Theme toggle for user preference

📐 System Architecture



🔄 WebSocket API Protocols

Protocol	Direction	Purpose
`session-history`	Client → Server	Request conversation history for session
`session-details`	Server → Client	Return session data with conversations
`conversation-message`	Client → Server	Send message for AI processing
`conversation-details`	Server → Client	Return updated conversation state
`conversation-delete`	Client → Server	Clear all conversations in session

📊 State Management

Actions

Action	Description
`setSelectedConversationId`	Set active conversation
`addMessage`	Add message to conversation
`setConversations`	Update all conversations
`setConversationHistory`	Update specific conversation
`deleteConversations`	Clear all conversations

🚀 Sprint Structure

Sprint 1: MVP/Prototype (5 Weeks)

Week	Phase	Deliverable
1	Frontend UI	React component structure
2	Express Server	WebSocket server setup
3	Sessions	Session management implementation
4	Server Messaging	Message routing logic
5	OpenAI Integration	ChatGPT API connection
—	**Sprint 1 Retrospective**	Review & lessons learned

Sprint 1 Goal: Functional MVP with core chat functionality

Sprint 2: Final Product (6 Weeks)

Week	Phase	Deliverable
6	Deployment	Production deployment
7	Architecture Refactor	Remove Express server dependency
8	UI Enhancements	Dark/Light mode toggle
9	Rich Text Support	Markdown rendering & code highlighting
10	Data Management	Export conversations & rename chats
11	Polish & Testing	Bug fixes, performance optimization
—	**Sprint 2 Retrospective**	Final review & demo

Sprint 2 Goal: Production-ready application with enhanced features

📦 Functional Requirements

1. **FR-01:** User can send text messages and receive AI responses
2. **FR-02:** User can create new conversations
3. **FR-03:** User can switch between existing conversations
4. **FR-04:** User can view conversation history
5. **FR-05:** User can delete all conversations
6. **FR-06:** Sessions persist across browser refreshes (IndexedDB)

🔒 Non-Functional Requirements

1. **NFR-01:** Response time < 3 seconds for AI replies
2. **NFR-02:** WebSocket connection handles reconnection gracefully
3. **NFR-03:** IndexedDB storage supports offline history viewing
4. **NFR-04:** Application is responsive on desktop and mobile
5. **NFR-05:** Secure API key management for OpenAI credentials

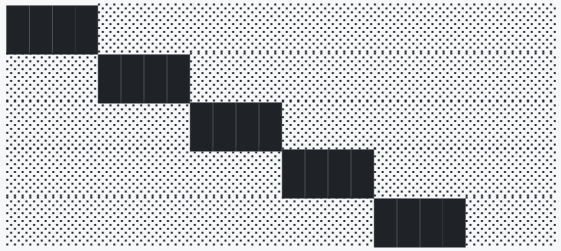
⚠️ Risks & Mitigations

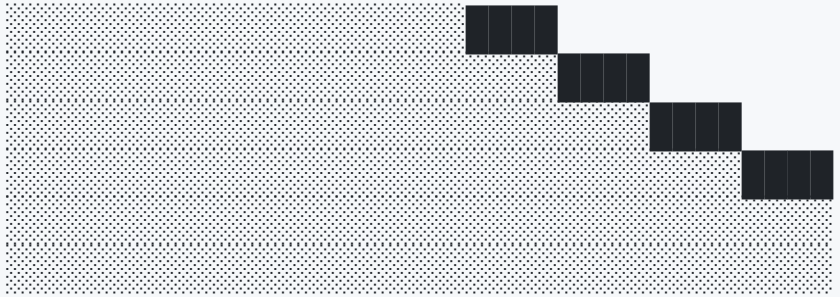
Risk	Impact	Mitigation
OpenAI API rate limits	High	Implement request throttling
WebSocket disconnection	Medium	Auto-reconnection logic
IndexedDB storage limits	Low	Implement data cleanup policies
API key exposure	High	Server-side API calls only

Timeline Overview

```
                    SPRINT 1 (5 Weeks)
                   MVP / Prototype
Week 1: Frontend UI
Week 2: Express Server
Week 3: Sessions
Week 4: Server Messaging
Week 5: OpenAI Integration
        ▼ Sprint 1 Retrospective ▼

                    SPRINT 2 (6 Weeks)
                   Final Product
Week 6:  Deployment
Week 7:  Refactor
Week 8:  UI Enhancements
Week 9:  Rich Text
Week 10: Data Management
Week 11: Polish & Testing
        ▼ Sprint 2 Retrospective ▼
```





✅ Success Criteria

Sprint 1 (MVP)

- [ ] Functional chat interface with real-time messaging
- [ ] Successful OpenAI API integration
- [ ] Persistent conversation storage with IndexedDB

Sprint 2 (Final)

- [ ] Deployed and accessible application
- [ ] Clean architecture after server refactor
- [ ] Dark/Light mode implemented
- [ ] Markdown & code highlighting working
- [ ] Export and rename conversation features complete