NeuraChat — Product Requirements Document (PRD)

1. Product Overview

Product Name: NeuraChat
Category: AI-Powered Real-Time Messaging Platform
Prepared By:

- Anees Hamid – 23L-0671
- Saad Zafar – 23L-0829
- Qatada – 23L-0998
- Ezaan – 23L-0676
  Instructor: Zeeshan Nazar
  Institution: FAST NUCES, Lahore

1.1 Product Summary

NeuraChat is a next-generation messaging application that merges **real-time communication** with **AI-powered enhancements**. It goes beyond conventional messaging apps by offering:

- Real-time AI grammar correction and message summarization
- Built-in AI assistant chat interface for productivity tasks
- WebRTC-based voice and video calling
- Secure end-to-end encrypted messaging via Signal Protocol

The product aims to redefine communication efficiency and intelligence in a unified platform.

---

2. Problem Statement

Current messaging apps (e.g., WhatsApp, Signal, Telegram) prioritize reliability and speed but lack:

- Integrated AI for message clarity, summarization, and suggestion
- Productivity-enhancing AI assistants
- Unified real-time text + voice + video system within one experience

NeuraChat bridges this gap by embedding AI-driven assistance into the messaging workflow.

---

3. Goals and Objectives

Goal	Description	Deliverable
G1: Real-time Chat	Build a secure real-time chat system using Next.js (frontend), Node.js + WebSockets (backend).	Chat module with message sync via Supabase
G2: AI Text Enhancement	Integrate grammar correction, text summarization, and message suggestions.	AI integration with OpenAI / local NLP models
G3: Voice & Video Calling	Implement WebRTC-based communication.	Call interface integrated in chat UI
G4: AI Assistant Chat	Provide a dedicated AI chatbot for productivity.	AI agent module with context memory
G5: Authentication & Security	Implement end-to-end encryption and user auth.	Signal Protocol + Supabase Auth
G6: Scalability & Performance	Ensure seamless experience under load.	Tested on simulated multi-user environment

---

4. Target Users

User Type	Description	Needs
Students/Professionals	Communicate effectively, correct grammar, summarize info	AI-assisted chat
Remote Teams	Quick message summaries and real-time calls	Collaboration efficiency
General Users	Privacy-focused daily communication	Fast, secure, smart messaging

---

5. Scope of Product

5.1 In-Scope Features

- Real-time chat using WebSockets
- Authentication via Supabase
- AI summarizer + grammar corrector (integrated in chat input/output)
- Dedicated AI assistant interface
- Voice and video calling using WebRTC
- Message history and media sharing
- Typing indicators, read receipts, online presence
- Encrypted data storage and communication

5.2 Out-of-Scope (Phase 1)

- Offline peer-to-peer file transfer
- Group video conferencing beyond 1:1
- End-user analytics dashboard

---

6. Functional Requirements

ID	Functionality	Description	Priority
FR1	User Registration & Login	Auth via Supabase (email/password, OAuth).	High
FR2	Real-Time Messaging	Send/receive messages instantly via WebSockets.	High
FR3	AI Grammar Correction	On message send, correct grammar automatically.	High
FR4	AI Text Summarization	On message receive, generate a short summary snippet.	Medium
FR5	AI Assistant Chat	Separate tab for chatting with built-in AI agent.	Medium
FR6	WebRTC Calling	1:1 video/voice call integrated in chat view.	High
FR7	File/Media Sharing	Send images, documents, audio clips.	Medium
FR8	Message Encryption	Implement end-to-end encryption using Signal Protocol.	High
FR9	Notifications	Real-time push notifications for new messages.	Medium

---

7. Non-Functional Requirements

Type	Requirement	Description
Performance	Response time < 200ms for chat ops	Ensure low-latency communication
Scalability	Support 1000+ concurrent users	Horizontally scalable backend

Security	End-to-end encryption	Confidentiality + integrity
Reliability	99.5% uptime during load testing	Supabase SLA + WebSocket resilience
Usability	Intuitive modern UI	Minimalist and responsive design (Next.js + Tailwind)
Maintainability	Modular code architecture	Separation of concerns (frontend/backend/AI modules)

---

8. System Architecture

Frontend:

- **Next.js + TailwindCSS** for responsive UI
- Uses **Supabase client** for auth and real-time updates

Backend:

- **Node.js + Express** for REST APIs and AI microservices
- **WebSockets** for real-time message sync
- **Signal Protocol** for encryption

AI Layer:

- **NLP Models (T5, BERT, GPT)** integrated through REST endpoints
- **AI Agent Service** for user queries and summarization

Database:

- **Supabase (Postgres)** with real-time sync channels
- Tables: Users, Chats, Messages, Media, Calls

Communication Layer:

- **WebRTC** for voice/video
- STUN/TURN servers for P2P connections

---

9. UX/UI Overview

9.1 Key Screens

1. **Login / Signup Page**
2. **Chat Dashboard**
3. **Conversation Window (AI Summary toggle)**
4. **AI Assistant Tab**
5. **Call Screen (Voice/Video)**
6. **Settings / Profile Page**

9.2 Figma Deliverable

- Wireframes for all screens
- User flow: Login → Chat → AI Assist → Call → Settings

---

10. Success Metrics

Metric	Target
Message delivery latency	< 150ms

AI summary generation	< 3 seconds
Average user retention	≥ 80% in closed beta
Crash-free sessions	≥ 98%
Call connection success rate	≥ 90%

---

11. Risks & Mitigation

Risk	Description	Mitigation
AI model latency	Slow API responses	Use lightweight local summarization for short messages
Network issues	Call drop or lag	Use adaptive bitrate WebRTC and retry logic
Model bias/errors	AI-generated mistakes	Provide manual toggle and disclaimer
Privacy	User data exposure	Apply encryption and anonymized logging

---

12. Deliverables (for Jira Epics)

Phase	Epic	Objectives	Key Deliverables	Milestone

🧭 **Phase 1 — Requirements & MVP Scoping (Days 1–4)**	Requirements and Planning	Establish clear MVP scope, prioritize features, finalize architecture outline.	• ✅ Requirements Document (PRD / Google Doc / PDF)• ✅ User Stories with Acceptance Criteria (Jira Backlog)• ✅ Defined Non-Functional Requirements (Performance / Security / Offline Behavior)• ✅ Jira Project Configured (Epics, labels, components)• ✅ GitHub Repository Initialized (with branch naming convention + README)	Project Kickoff + Backlog Approval

🎨 **Phase 2 — UI/UX Design in Figma (Days 5–10)**	Figma UI/UX Design & Prototyping	Design full Figma prototype aligned with MVP features.	• ✅ Clickable Figma Prototype (Wireframes + Hi-Fi Mockups)• ✅ Component Spec Sheet (Design Tokens / Typography / Color System)• ✅ Accessibility Checklist (Contrast + Keyboard Flow)• ✅ Design Sign-off Meeting Minutes / Approval Sheet	Design Finalized for Dev Handoff
🗄️ **Phase 3 — Database Design (Supabase) (Days 11–15)**	Database & System Architecture	Model and implement Supabase schema with row-level security and real-time channels.	• ✅ ER Diagram (Lucidchart / Draw.io)• ✅ Database Schema DDL (SQL file)• ✅ Supabase Project Setup (with RLS + Policies)• ✅ Realtime Channel Plan + Testing Report• ✅ API Contract Draft (OpenAPI Spec)	Backend Ready to Connect

⚙️ **Phase 4 — Backend Development (Days 16–24)**	Backend Core & Realtime Services	Build REST + WebSocket backend integrated with Supabase.	• ✅ REST API Documented (Postman / OpenAPI)• ✅ WebSocket Event Schema + Realtime Handlers• ✅ Auth + Middleware Implemented (Supabase Token Validation)• ✅ AI Endpoint Stubs (Mock Summarizer / Grammar API)• ✅ Unit Test Coverage Report• ✅ Backend Deployed (Vercel Serverless Functions)	Backend Feature-Complete (API Contracts Stable)

💻 **Phase 5 — Frontend Development (Days 25–33)**	Frontend Implementation & Integration	Develop complete frontend (Next/Vite + Tailwind + Supabase integration).	• ✅ Connected Frontend (Vercel Preview Deploy)• ✅ Auth Flow + Routing (Protected Routes)• ✅ Chat UI (Threads, Composer, Status Indicators)• ✅ Realtime Messaging Integrated with WebSocket• ✅ Offline Sync via IndexedDB/Localforage• ✅ Component Library (Storybook Optional)	Functional Chat App MVP

🧠 **Phase 6 — AI Integration (Days 34–37)**	AI Summarizer & Grammar Assistant	Integrate AI summarization and grammar correction into chat flow.	• ✅ AI Adapter Connected to OpenAI / Local Endpoint• ✅ Summaries Generated and Cached in DB• ✅ Grammar Correction Inline Suggestions Functional• ✅ AI Agent Chat Thread Working with Streaming Response• ✅ Settings Toggle for AI Features	AI Features Complete
📞 **Phase 7 — WebRTC Calling (Days 38–41)**	WebRTC 1:1 Calling Feature	Implement signaling and peer connection for voice/video calls.	• ✅ Signaling via WebSocket (Offer/Answer/ICE Schema)• ✅ Client Peer Connection (getUserMedia + Call Modal UI)• ✅ 1:1 Call Tested in Staging (LAN + Remote)• ✅ Call Session Logs Saved in Supabase• ✅ Error Handling and Reconnect Flow	Voice Call MVP Functional

🧪 **Phase 8 — Testing & QA (Days 42–45)**	Final Testing & Handover	Validate end-to-end flows, finalize docs and demo build.	• ✅ Final Staging Build (Vercel Deploy)• ✅ QA Test Report (Functional + Integration)• ✅ Performance and Security Validation (RLS/Auth Checks)• ✅ Documentation Bundle (README + Setup Guide + Diagrams)• ✅ Demo Video / Presentation Slides• ✅ Final Project Report PDF	Project Ready for Submission / Demo

---

13. Future Enhancements

- Multi-language grammar support
- Group voice/video calls
- Chatbots marketplace
- Offline LAN messaging
- AI emotion detection for messages