Use Case Descriptions

Use Case #1

Field	Detail
Identifier	UC-1
Name	**User Authentication**
Summary	The process by which a user registers a new account or logs into an existing account. This core function leverages **Supabase** as the primary database and authentication service to handle secure user authentication.
Priority	High
Actors	User
Pre-condition(s)	The user has access to the Next.js frontend of the NeuraChat application.
Post-condition(s)	The user is successfully authenticated and logged in, allowing access to application features, including real-time messaging and the built-in AI assistant. The system has leveraged **Supabase** for secure user synchronization.

Typical Course of Action (Login Scenario)

S#	Actor Action	System Response
1	The user attempts to access the NeuraChat application.	
2		The system presents the login/authentication interface.
3	The user enters existing credentials (e.g., email and password) and submits the login request.	

4		The system communicates the credentials to the **Supabase** authentication service.
5		**Supabase** validates the user credentials and returns a success response.
6		The system establishes a secure session and grants the user access to the main application interface.

Alternate Course of Action (Invalid Credentials)

S#	Actor Action	System Response
3a	The user enters incorrect or invalid credentials.	
4a		The system communicates the credentials to **Supabase**.
5a		**Supabase** returns an authentication failure response.
6a		The system displays an error message (e.g., "Invalid credentials") and prompts the user to try again.

Use Case #2

Field	Detail
Identifier	UC-2
Name	**Send Real-Time Text Message**
Summary	The user initiates and successfully transmits a standard text message to another user or group in real-time. This fundamental communication feature is facilitated by the **robust Node.js backend**, utilizing **WebSockets** for immediate data transfer.
Priority	High
Actors	User (Sender), Recipient User(s)
Pre-condition(s)	1. The sender is successfully authenticated (UC-1). 2. The sender has an active chat session or group chat open.
Post-condition(s)	1. The message is transmitted successfully via **WebSockets**. 2. The recipient receives the message in real-time. 3. The message is stored and synchronized using the platform's backend services.

Typical Course of Action

S#	Actor Action	System Response
1	The user types the message content into the chat input field.	

2	The user selects or triggers the send action.	
3		The system sends the message data to the **Node.js backend**.
4		The **Node.js backend** processes the message and uses **WebSockets** to immediately push the message to the target recipient(s).
5		The system confirms the message delivery status in the sender's chat view.
6		The recipient's application receives and displays the message in real-time.

Alternate Course of Action (Temporary Connectivity Loss)

S#	Actor Action	System Response
3a	The user selects or triggers the send action while experiencing poor connectivity.	
4a		The system detects that the real-time **WebSocket** connection to the **Node.js backend** is unavailable or timed out.
5a		The system leverages its **offline resilience capabilities** by storing the message locally and displaying a "pending" status (as described in UC-16).

6a	Once connectivity is restored, the system automatically transmits the pending message to the **Node.js backend**.

Use Case #3

Field	Detail
Identifier	UC-3
Name	**Share Multimedia Content**
Summary	The user shares files, images, or other multimedia content within a chat conversation. This capability is included as one of the basic features of the messaging platform.
Priority	High
Actors	User (Sender), Recipient User(s)
Pre-condition(s)	1. The sender is successfully authenticated (UC-1). 2. The sender has an active chat session open.
Post-condition(s)	1. The multimedia content is securely uploaded to the backend (leveraging scalable cloud storage). 2. A reference to the content is successfully transmitted and displayed in the chat conversation.

Typical Course of Action

S#	Actor Action	System Response
1	The user taps/clicks the attachment icon to initiate sharing multimedia content (files, images, etc.).	
2	The user selects the desired multimedia file from their device.	

3	The user confirms the file selection and initiates the send process.	
4		The system initiates the upload of the multimedia content to the scalable cloud storage solution (e.g., Supabase storage).
5		Upon successful upload, the system sends a message reference via the robust Node.js backend to the recipient(s).
6		The recipient’s application receives and displays the multimedia content (e.g., a preview or download option) in real-time.

Alternate Course of Action (Upload Failure)

S#	Actor Action	System Response
3a	The user initiates sending a multimedia file.	
4a		The system attempts the upload but encounters an error (e.g., connection drop or file size exceeding limits defined in the scalable cloud storage).
5a		The system terminates the upload and displays a notification to the user indicating the failure (e.g., "Upload failed").

Use Case #4

Field	Detail
Identifier	UC-4
Name	**Initiate Real-Time Audio Call**
Summary	The user starts a real-time voice call with another user. This feature is powered by **WebRTC integration**, which allows for seamless, peer-to-peer voice communication integrated directly into the chat system.
Priority	High
Actors	User (Caller), Recipient User (Callee)
Pre-condition(s)	1. The caller and callee are successfully authenticated (UC-1). 2. The application is designed to support real-time voice calling using WebRTC.
Post-condition(s)	1. A secure, real-time, peer-to-peer audio connection is established between the caller and callee. 2. The users are actively engaged in an audio conversation.

Typical Course of Action

S#	Actor Action	System Response
1	The caller selects a contact and initiates the real-time audio call function.	

2		The system uses the **Node.js backend** to send a call notification (signaling) to the callee.
3	The callee receives the notification and accepts the incoming audio call.	
4		The system initiates the **WebRTC** protocol negotiation to establish a direct, peer-to-peer audio stream between the two users.
5		The system establishes the connection, and the users begin real-time voice communication, seamlessly integrated with the chat system.

Alternate Course of Action (WebRTC Network Failure)

S#	Actor Action	System Response
3a	The callee accepts the call, but one or both users are in a low-bandwidth environment.	
4a		The system attempts the **WebRTC** negotiation, but the connection process times out or fails to establish a stable peer-to-peer stream due to poor network quality.

5a	The system terminates the call attempt and displays a warning to the users indicating potential latency or a dropped call due to low-bandwidth limitations.

Use Case #5

Field	Detail
Identifier	UC-5
Name	**Initiate Real-Time Video Call**
Summary	The user initiates a real-time video call with another user. This core functionality is powered by **WebRTC integration**, extending the communication experience beyond text and multimedia messages. It is integrated seamlessly with the chat system.
Priority	High
Actors	User (Caller), Recipient User (Callee)
Pre-condition(s)	1. The caller and callee are successfully authenticated (UC-1). 2. The system has implemented real-time video calling using WebRTC.
Post-condition(s)	1. A secure, real-time, peer-to-peer video connection is established between the users via WebRTC. 2. The users are actively engaged in a video conversation.

Typical Course of Action

S#	Actor Action	System Response
1	The caller selects a contact and initiates the real-time video call function.	

2		The system sends a call notification (signaling) to the callee, utilizing the robust Node.js backend for real-time interactions.
3	The callee receives the notification and accepts the incoming video call.	
4		The system initiates the **WebRTC** negotiation process to establish a direct, peer-to-peer video stream.
5		The system establishes the connection, and the users begin real-time video communication, seamlessly integrated with the chat application.

Alternate Course of Action (Network Quality Limitation)

S#	Actor Action	System Response
3a	The callee accepts the call, but one or both users are in a low-bandwidth or low-connectivity environment.	
4a		The system attempts the **WebRTC** connection, but the feature is limited by network quality. The connection process fails to establish a stable peer-to-peer stream or times out.

5a	The system terminates the video call attempt and displays a warning to the users indicating the call was dropped due to network constraints or potential latency issues.

Use Case #6

Field	Detail
Identifier	UC-6
Name	**Correct Grammar in Outgoing Message**
Summary	The user types a message, and the **AI-based grammar correction system** automatically improves the **message quality**. This system is integrated in real-time to assist users in writing clear and professional messages.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user is actively typing a message in a chat session. 3. The **AI-based grammar correction system** is initialized and monitoring text input.
Post-condition(s)	The outgoing message draft has been reviewed and optionally improved by the AI system.

Typical Course of Action (Successful Correction)

S#	Actor Action	System Response
1	The user types a text message containing grammatical or syntactical errors.	
2		The system captures the text input in real-time for processing.

3		The system feeds the input to the integrated **AI-based grammar correction system**.
4		The AI model identifies the errors and generates a corrected version of the message, thereby automatically improving the message quality.
5		The system displays the grammatically corrected text or highlights suggested changes to the user in the input field.
6	The user accepts the correction (or sends the automatically corrected message).	

Alternate Course of Action (AI Limitation)

S#	Actor Action	System Response
3a	The user types a highly complex or idiomatic phrase that challenges the system.	
4a		The system attempts correction, but the **accuracy may vary** because the AI summarization and grammar correction depend on pre-trained models. The system fails to identify the error or suggests an inappropriate correction.
5a		The system displays the original text or a poor correction.

6a The user disregards the AI suggestion and manually adjusts the message content before sending.

Use Case #7

Field	Detail
Identifier	UC-7
Name	**Enhance Message Clarity**
Summary	The user utilizes **NLP tools** to **enhance the clarity and professionalism of a message before sending it**. This objective is a core feature aimed at assisting users in writing clear, concise, and professional messages.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user is actively typing a message draft in a chat session.
Post-condition(s)	The message draft has been reviewed and optionally modified by the **NLP tools** to enhance its clarity and professionalism.

Typical Course of Action

S#	Actor Action	System Response
1	The user types a message draft into the chat input field.	
2	The user explicitly triggers the message enhancement function (or the system processes it in real-time).	

3		The system feeds the draft message to the integrated **NLP tools**.
4		The **NLP tools** analyze the structure and vocabulary to generate suggestions designed to **enhance the clarity and professionalism of the message**.
5		The system displays the suggested, enhanced version of the message to the user for review.
6	The user accepts the suggestion, overwriting the original draft with the enhanced text.	
7	The user sends the now enhanced and clarified message (UC-2).	

Alternate Course of Action (User Rejects Enhancement)

S#	Actor Action	System Response
4a		The system generates a suggested enhancement that does not align with the user's intended tone or meaning.
5a		The system displays the suggested enhancement (Step 5 of the typical course).
6a	The user reviews the suggestion and rejects the change, choosing to keep the original text.	

7a The user sends the original message without NLP enhancement (UC-2).

Use Case #8

Field	Detail
Identifier	UC-8
Name	**Summarize Outgoing Message**
Summary	The user employs the **AI-based real-time text summarizer** to help write **clear and concise outgoing messages**. This feature integrates intelligent support into the chat flow to assist users in optimizing their communications.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user has typed a message draft that is lengthy enough to require condensation. 3. The **AI-based real-time text summarizer** is active.
Post-condition(s)	The user's outgoing message draft is reviewed and transformed into a shorter, more concise summary, improving message quality.

Typical Course of Action (Successful Summarization)

S#	Actor Action	System Response
1	The user types a detailed or lengthy message draft in the chat input field.	
2	The user triggers the summarization function (or the system detects the need for summarization automatically).	

3		The system sends the message draft to the integrated **AI-driven text summarizer**.
4		The system processes the text and generates a concise version of the outgoing message.
5		The system displays the suggested summarized text to the user for review in place of the original draft.
6	The user accepts the summarized version.	
7	The user sends the condensed, concise message (UC-2).	

Alternate Course of Action (Inaccurate Summary)

S#	Actor Action	System Response
3a	The user submits a complex draft to the summarizer.	
4a		The system processes the text, but since the **AI summarization will depend on pre-trained models**, the **accuracy may vary**. The resulting summary is either inaccurate or loses critical nuance.
5a		The system displays the low-quality summary to the user.

6a The user reviews the unsatisfactory summary and rejects the change, choosing to keep the original text.

7a The user manually edits the original message or sends the original text without AI summarization (UC-2).

Use Case #9

Field	Detail
Identifier	UC-9
Name	**View Automatic Summary of Incoming Message**
Summary	The user receives a long message, and the **AI-driven text summarizer automatically condenses it into a short, readable summary**. This feature provides users with **quick insights without the need to read lengthy texts**.
Priority	High
Actors	User (Recipient)
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user has received an incoming message exceeding a predefined length threshold. 3. The **AI-driven text summarizer** is active to **condense long incoming messages**.
Post-condition(s)	The recipient is presented with the original message and a concise, readable summary of its content.

Typical Course of Action (Successful Summarization)

S#	Actor Action	System Response
1	The user receives a notification for a new, long incoming message.	
2		The system receives the incoming message via the robust Node.js backend (UC-2).

3		The system recognizes the length of the message and sends it to the integrated **AI-driven text summarizer**.
4		The summarizer processes the text and generates a **short, readable summary**.
5		The system displays both the original message and the automatically generated summary to the user, providing **quick insights**.
6	The user reviews the summary to understand the message's main points.	

Alternate Course of Action (Summary Accuracy Varies)

S#	Actor Action	System Response
4a		The system attempts summarization, but because **AI summarization will depend on pre-trained models**, the **accuracy may vary**. The resulting summary is either nonsensical or misses the main topic.
5a		The system displays the inaccurate summary alongside the original message.
6a	The user determines the summary is unhelpful or inaccurate.	

7a The user ignores the summary and reads the original long text to obtain the complete context.

Use Case #10

Field	Detail
Identifier	UC-10
Name	**Generate Long Message Drafts**
Summary	The user leverages **NLP tools to aid in writing long and detailed messages based on provided user information**. This advanced feature helps users quickly compose complex or comprehensive communications.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user is actively using the chat interface. 3. The system's **NLP tools** are initialized and available to **aid in writing long and detailed messages**.
Post-condition(s)	A comprehensive, detailed message draft is generated by the AI and placed into the user's input field for final review and editing.

Typical Course of Action

S#	Actor Action	System Response
1	The user enters key information, keywords, or a prompt for the desired long message (e.g., "Draft a status update on the project delay due to supply chain issues").	
2	The user triggers the long message generation function.	
3		The system sends the user input and context to the integrated **NLP tools**.

4		The NLP tools process the request and generate a **long and detailed message based on provided user information**.
5		The system inserts the resulting detailed draft into the message input field.
6	The user reviews, makes minor edits, and sends the detailed message (UC-2).	

Alternate Course of Action (Low Accuracy Draft)

S#	Actor Action	System Response
3a		The system attempts to generate the message, but because AI generation depends on pre-trained models, the **accuracy may vary**. The resulting draft contains factual inaccuracies or misses the user's intended tone.
4a		The system presents the low-quality draft in the input field.
5a	The user reviews the draft and determines it is unsuitable for sending.	
6a	The user either manually edits the entire draft heavily or clears the draft and composes the message manually.	

Use Case #11

Field	Detail
Identifier	UC-11
Name	**Access Dedicated AI Agent Chat**
Summary	The user navigates to the **dedicated section designed for interacting with the built-in AI assistant**. This interface allows users to interact with the AI assistant for various tasks, including **information retrieval, writing help, and productivity support**.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The AI agent services are active and available in the application backend.
Post-condition(s)	The user is actively viewing the dedicated AI Agent Chat Interface, ready to send a query to the built-in AI assistant.

Typical Course of Action

S#	Actor Action	System Response
1	The user navigates to the application's main menu or feature selection panel.	
2	The user selects the option to access the dedicated AI Agent Chat interface.	
3		The system loads the **dedicated section designed for interacting with the built-in AI assistant**.

4		The system initializes the chat interface, displaying the AI agent's welcome message and the input field.
5	The user inputs a query related to information retrieval, writing help, or productivity support.	

Alternate Course of Action (AI Service Unavailable)

S#	Actor Action	System Response
2a	The user selects the option to access the dedicated AI Agent Chat interface.	
3a		The system attempts to connect to the AI agent service but receives an error or timeout response.
4a		The system displays a notification indicating that the built-in AI assistant is temporarily unavailable or offline.

Use Case #12

Field	Detail
Identifier	UC-12
Name	**Use AI Agent for Information Retrieval**
Summary	The user queries the **dedicated AI agent** to **retrieve specific information**. This interaction takes place within the dedicated AI agent chat interface.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user has accessed the dedicated AI Agent Chat interface (UC-11).
Post-condition(s)	The built-in AI assistant has processed the query and successfully delivered the requested information to the user.

Typical Course of Action

S#	Actor Action	System Response
1	The user types a request or query for specific information into the AI Agent Chat interface (e.g., "What are the latest stock market figures?").	
2	The user sends the query to the dedicated AI agent.	
3		The system feeds the query to the built-in AI assistant.
4		The AI agent processes the request and retrieves the specific information.

5		The system displays the retrieved information as a direct response in the chat interface.

Alternate Course of Action (Information Not Found/Internal Limitation)

S#	Actor Action	System Response
3a	The user submits a query to the AI agent.	
4a		The AI agent attempts to retrieve the information but is unable to find a relevant or verifiable answer, or the knowledge model lacks the necessary context.
5a		The system displays a polite error message or notification stating that the AI assistant could not fulfill the request or that the requested information is outside its scope.

Use Case #13

Field	Detail
Identifier	UC-13
Name	**Use AI Agent for Writing Assistance**
Summary	The user asks the **dedicated AI agent for general writing help (beyond grammar correction)**. This includes assistance with structuring, tone, drafting complex text, or generating ideas, leveraging the built-in AI assistant.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user has accessed the dedicated AI Agent Chat interface (UC-11).
Post-condition(s)	The built-in AI assistant has processed the query and successfully provided writing assistance, suggestions, or a draft to the user.

Typical Course of Action

S#	Actor Action	System Response
1	The user types a request for general writing help (e.g., "Draft an outline for a project proposal," or "Suggest a more professional tone for this paragraph").	
2	The user sends the request to the dedicated AI agent.	
3		The system feeds the query to the built-in AI assistant for writing help.

4		The AI agent generates and formats the requested writing assistance (e.g., a draft, an outline, or suggested alternative phrasing).
5		The system displays the AI agent's response in the chat interface.
6	The user copies or utilizes the provided writing assistance in their communication.	

Alternate Course of Action (Ambiguous Request)

S#	Actor Action	System Response
3a	The user submits a vague or highly abstract query for writing help.	
4a		The AI agent processes the request but cannot fulfill it precisely due to lack of specificity or the limitations of the pre-trained models.
5a		The system displays a request for clarification or provides a generic, unhelpful response (e.g., "Please provide more detail about the required document").
6a	The user refines the original request and resubmits the query.	

Use Case #14

Field	Detail
Identifier	UC-14
Name	**Use AI Agent for Productivity Support**
Summary	The user interacts with the **built-in AI agent** for tasks related to **productivity support**. This feature leverages the dedicated AI agent chat interface to assist users with organizational, scheduling, or general productivity tasks.
Priority	High
Actors	User
Pre-condition(s)	1. The user is successfully authenticated (UC-1). 2. The user has accessed the dedicated AI Agent Chat interface (UC-11).
Post-condition(s)	The AI agent has processed the request and executed or provided assistance for a productivity-related task.

Typical Course of Action

S#	Actor Action	System Response
1	The user types a request into the AI Agent Chat interface related to a productivity task (e.g., "Summarize my unread emails," or "Create a to-do list based on our recent chat").	
2	The user sends the request to the dedicated AI agent.	
3		The system feeds the query to the built-in AI assistant for productivity support.

4		The AI agent processes the request, potentially interfacing with other user data or internal models.
5		The system displays the AI agent's response, which fulfills the productivity task (e.g., a summarized list, a reminder confirmation, or a suggested schedule).

Alternate Course of Action (External Integration Required)

S#	Actor Action	System Response
3a	The user submits a query that requires integration with external tools (e.g., "Add an event to my Google Calendar").	
4a		The AI agent processes the request but detects it requires external authorization or an integration that is not currently scoped or available.
5a		The system displays a response indicating the limitation (e.g., "I can help you draft the text, but I cannot directly access or modify your external calendar.").
6a	The user refines the request to ask for a textual draft or outline instead.	

Use Case #15

Field	Detail
Identifier	UC-15
Name	**Ensure Secure Communication**
Summary	The system automatically secures user communications using the **Signal Protocol**, guaranteeing **end-to-end encryption**, **forward secrecy**, and **post-compromise security**. This is a key objective for securing communication.
Priority	Critical
Actors	System (Primary Executor), User (Beneficiary)
Pre-condition(s)	1. The users involved in the communication are successfully authenticated (UC-1). 2. A communication channel (text, audio, or video) is being established or is active. 3. The **Signal Protocol** implementation is initialized.
Post-condition(s)	All exchanged data is secured using the **Signal Protocol**, guaranteeing end-to-end encryption, forward secrecy, and post-compromise security.

Typical Course of Action (Sending a Secure Message)

S#	Actor Action	System Response
1	The sender drafts and initiates sending a message (related to UC-2).	
2		The system automatically encrypts the message data using the **Signal Protocol** on the sender's device.
3		The system transmits the encrypted message via the robust Node.js backend using WebSockets.

4		Upon receipt, the system automatically decrypts the message data using the **Signal Protocol** on the recipient's device.
5		The system displays the clear text message to the recipient, confirming the message maintained **end-to-end encryption**.

Alternate Course of Action (Key Management and Forward Secrecy)

S#	Actor Action	System Response
1a	A third party successfully compromises a temporary session key used in a previous conversation.	
2a		The system's implementation of the **Signal Protocol** prevents the compromised key from decrypting any messages sent *before* the key was compromised.
3a		The system regenerates session keys for subsequent communication attempts, thereby maintaining **forward secrecy** and **post-compromise security**.