生成AIの「悩み」を解決する人間参加型応答アプリケーション開発のための調査報告書

1. はじめに：生成AIと人間の協調

近年、生成AI（Generative AI）は目覚ましい発展を遂げ、テキスト生成、翻訳、要約、コード生成など、多岐にわたるタスクで人間のような能力を発揮しています[1]。しかし、その能力には限界も存在し、特定の状況下ではAIが「悩む」、すなわち最適な応答を生成できない、あるいは不確実性の高い応答しか返せない場面が見られます。

本報告書は、生成AIが「悩む」状況において、人間が適切な「受け答え」を提供することでAIの判断や応答生成を支援するアプリケーション（以下、「人間応答アプリ」）の開発を目的としています。このアプリケーションは、AIの能力を補完し、より信頼性が高く、文脈に適したAIの活用を促進することを目指します。

本報告書では、まずユーザーが言及した「MCP」の概念を整理し、次にAIが悩む具体的な状況、有効な人間の応答形式、関連する既存システム、アプリケーションに必要な機能、技術的連携方法、推奨される技術スタック、そしてプライバシーとセキュリティに関する考慮事項について、詳細な調査結果を提示します。

1.1. 「MCP」の概念整理：Model Context Protocol

ユーザーが言及した「MCP」は、生成AIの文脈において、特定の技術プロトコルである**Model Context Protocol (MCP)** を指している可能性が高いと考えられます。これは一般的な概念ではなく、Anthropic社によって提唱され、オープンソース化された特定の標準規格です[2]。

MCPは、AIモデル（特にLLM）が外部のツールやサービス、データソースと安全かつ標準化された方法で通信するためのプロトコルとして設計されています[2]。これは、AIモデルが自身の内部知識だけでは対応できないタスク（例：最新情報の取得、API連携、データベースクエリ、ファイル操作）を実行可能にすることを目的としています[2]。

MCPは、AIアプリケーション（ホスト）、MCPクライアント、MCPサーバーという3つの主要コンポーネントから構成されるクライアント・サーバーアーキテクチャを採用しています[2]。

- ホスト **(Host):** ユーザーが直接対話するAI駆動アプリケーション（例: IDE、チャットインターフェース）[2]。
- クライアント **(Client):** ホスト内に存在し、特定のMCPサーバーとの1対1の接続を維持・管理する仲介役[2]。
- サーバー **(Server):** 外部のツール、データソース、APIへのアクセスを提供するコンポーネント。Webサービスだけでなく、ローカルファイルシステムへのアクセスも提供可能[2]。

MCPは、AIと外部システム間の連携を「ツール（Tools）」「リソース（Resources）」「プロンプト（

Prompts）」という3つのプリミティブ（基本要素）で標準化します [3]。

- ツール **(Tools):** AIが実行可能な関数（API呼び出し、DBクエリなど）[3]。
- リソース **(Resources):** AIが参照できる構造化データ（ファイル、DBレコード、APIレスポンスなど）[3]。
- プロンプト **(Prompts):** 再利用可能な指示テンプレート [3]。

MCPの利点は、開発者がAIモデルごとに、あるいは連携したいツールごとにカスタム統合を構築する必要がなくなり、標準化されたプロトコルに対応するだけで多様な外部システムとの連携が可能になる点です [2]。これにより、開発効率の向上、AIの能力拡張、エコシステムの成長が期待されます [2]。

当初のユーザーの意図は「AIが悩んだときに人間が助ける仕組み」全般を指していたかもしれませんが、技術的な文脈ではMCPはこの特定のプロトコルを指します。本報告書で提案する人間応答アプリは、MCPを利用するAIシステムと連携することも可能ですが、MCP自体が直接的に「AIの悩みに対する人間の応答」を標準化するものではありません。むしろ、MCPはAIが外部ツール（人間応答アプリを含む可能性もある）と連携するための「手段」の一つとなり得ます。

2. 生成AIが「悩む」状況の特定

生成AIは多くのタスクで高い能力を発揮しますが、特定の状況下では最適な応答を生成することが困難になります。これらの「悩み」の状況を理解することは、人間応答アプリがどのような場面で価値を提供できるかを明確にする上で不可欠です。

2.1. 曖昧な指示や文脈不足

AIは、与えられた指示（プロンプト）や文脈に基づいて応答を生成します。指示が曖昧であったり、必要な背景情報が不足していたりすると、AIは何を求められているのかを正確に理解できず、意図しない、あるいは的外れな応答を生成する可能性があります [16]。例えば、「ブログ記事を書いて」という指示だけでは、トピック、対象読者、文体などが不明確なため、AIは汎用的な内容しか生成できません [16]。

AIは文脈を考慮しますが [16]、その文脈が不足している場合や、複数の解釈が可能な曖昧な表現が含まれている場合 [17]、AIはどの解釈に基づいて応答すべきか判断に迷うことがあります。

2.2. 倫理的な判断やバイアス

AIモデルは、学習データに含まれるバイアスを継承・増幅してしまう可能性があります [21]。これにより、性別、人種、その他の属性に基づいて差別的な内容やステレオタイプに基づいた応答を生成してしまうリスクがあります。例えば、過去の採用データに基づいて学習したAIが、特定の性別を不当に評価する可能性があります [25]。

また、AI自身には倫理観がないため、社会的に許容されないコンテンツや、法的に問題のあるコンテンツを生成してしまう可能性も指摘されています [21]。さらに、自動運転車における「トロッコ問題」のように、避けられない危害が発生する場合にどちらを優先すべきかといった複雑な倫理的ジレンマに対して、AIが自律的に判断を下すことの是非や責任の所在も大きな課題です [21]。AIの意思決定プロセスが不透明（ブラックボックス）であることも、これらの問題を複雑にしています [21]。

2.3. 創造的な発想や新規性

生成AIは、既存のデータパターンに基づいて新しいコンテンツを生成することに長けていますが、真に独創的で、既存の枠を超えたアイデアを生み出すことは苦手とする場合があります [28]。特に、完全に新しい概念の創出や、深い芸術性、人間特有の感性が求められるタスクにおいては、AIの生成物は定型的で深みに欠けることがあります [21]。

AIはアイデア出しの「たたき台」やインスピレーションの源としては有用ですが [19]、創造的なプロセスにおける最終的な判断や洗練には、人間の感性や経験が依然として重要です。クリエイティブな作業において「行き詰まり（Creative Block）」を感じた際に、AIをブレインストーミングのパートナーとして活用するアプローチも提案されています [33]。

2.4. 事実確認（ファクトチェック）の困難さ

生成AI、特に大規模言語モデル（LLM）は、「ハルシネーション（幻覚）」と呼ばれる、事実に基づかないもっともらしい情報を生成してしまう問題が知られています [20]。これは、学習データの不完全さや不正確さ、モデルが知らないことを認めるよりもっともらしい回答を生成しようとする傾向などが原因とされています [38]。

AIは存在しない研究論文を引用したり、歴史的事実を誤って述べたりすることがあります [21]。特に、最新の情報（学習データのカットオフ日以降の出来事）や、専門性の高い分野、ニッチな分野の情報に関しては、AIの知識は不完全であり、誤った情報を生成するリスクが高まります [39]。AIによるファクトチェックの試みもありますが、文脈のニュアンス（皮肉や風刺など）の理解不足や、進化する誤情報への追従、バイアスの問題など、依然として課題が多く、人間の判断が不可欠な場面が多く存在します [38]。

2.5. その他の限界

上記以外にも、LLMには以下のような限界や課題が指摘されています。

- 常識推論の困難さ: 人間にとっては自明な常識に基づいた推論が苦手な場合があります [1]。
- 計算能力: 複雑な算術計算や論理演算を苦手とすることがあります [1]。
- 知識の固定化: 特定の時点までのデータで学習されているため、それ以降の知識は基本的に持っていません [39]。
- データ依存性: 学習データの質や量、偏りに性能が大きく左右されます [39]。

- コスト: 大規模モデルの学習と運用には膨大な計算リソースとコストが必要です [1]。

これらの「悩み」の状況は、AI単独では最適な解決策を見つけるのが難しい場面であり、人間応答アプリが介入し、AIを支援することで大きな価値を発揮できる領域と言えます。AIの「悩み」は単純な正誤の問題だけでなく、文脈理解、倫理観、創造性、事実性といった多岐にわたる側面を含んでおり、人間による支援のあり方も、それぞれの状況に合わせて設計する必要があります。

3. 人間による効果的な「受け答え」の形式

AIが「悩む」状況において、人間がどのような「受け答え」を提供すればAIの助けとなるのでしょうか。その形式は、AIが直面している問題の種類や、求める支援のレベルによって異なります。

3.1. 曖昧さの解消：明確化と具体化

AIが曖昧な指示に悩んでいる場合、人間は指示をより具体的かつ明確にすることで支援できます [16]。

- 選択肢の提示: AIが複数の解釈に迷っている場合、人間が解釈の選択肢を提示し、AIに進むべき方向を示すことができます。
- 自由記述による補足: 指示の意図や背景情報、制約条件などを自由記述で補足説明することで、AIの文脈理解を助けます [16]。例えば、「ターゲット層は20代女性」「文体はカジュアルに」「文字数は500字以内」といった具体的な情報を追加します [19]。
- 具体例の提供: 求めるアウトプットの具体的な例を示すことで、AIはより明確な目標を持ってタスクに取り組むことができます [16]。
- 質問による明確化: AIが自ら曖昧な点を質問し、人間がそれに答える形式も考えられます。プロンプトエンジニアリングのテクニックとして、AIに不明点を質問させる指示を含めることも有効です [17]。

3.2. 倫理的判断の指針：ガイダンスと価値判断

倫理的なジレンマに直面した場合、AIは規範的な判断を下すことができません。人間は倫理的な指針や価値判断を提供する必要があります。

- 規範やルールの提示: 適用されるべき倫理規範、法的要件、社内ポリシーなどを提示し、AIが判断の拠り所とできるようにします。
- 優先順位付け: 複数の価値が対立する場合（例：プライバシー保護 vs 公益性）、人間が状況に応じた優先順位を示します。
- 承認/拒否: AIが提案した行動や応答が倫理的に問題ないか人間がレビューし、承認または拒否します [45]。これは特に、AIが自律的に行動する可能性がある場合に重要です [21]。
- バイアスの指摘と修正: AIの応答にバイアスが含まれている場合、人間がそれを指摘し、

修正の方向性を示します [21]。

3.3. 創造性の刺激：アイデア提供とフィードバック

AIが創造的なタスクで行き詰まっている場合、人間は新たな視点やインスピレーションを提供できます。

- アイデアの壁打ち: AIが生成したアイデアに対して人間がフィードバックを与えたり、別の角度からのアイデアを提示したりすることで、発想を深めます [19]。
- 制約の変更: 異なる制約条件（例：異なるテーマ、文体、形式）を提示することで、AIに新たな方向性を探らせます。
- 関連情報の提供: 関連する作品、事例、キーワードなどを提供し、AIの「発想の種」を増やします [33]。
- 人間による編集・洗練: AIが生成した下書きを人間が編集し、より洗練された、あるいは人間味のある表現に仕上げます [19]。

3.4. 事実確認の支援：情報源の提示と検証

AIが不確かな情報やハルシネーションを生成した場合、人間は事実確認を支援します。

- 信頼できる情報源の提示: AIに対して、参照すべき信頼性の高い情報源（特定のウェブサイト、データベース、文献など）を指定します [20]。
- 情報の検証と修正: AIが生成した情報の正誤を人間が検証し、誤りがあれば訂正します [20]。これは特に、医療や法律など、正確性が極めて重要な分野で不可欠です [28]。
- 根拠の要求: AIに対して、生成した情報の根拠を示すよう要求し、その妥当性を人間が評価します [20]。
- 「不明」の許容: AIが確信を持てない場合に、「不明」または「わからない」と回答するように指示することも、誤情報のリスクを低減する上で有効です [20]。

これらの応答形式は、AIの「悩み」の種類に応じて使い分けることが重要です。例えば、曖昧さには明確化、倫理問題には指針、創造性の壁には刺激、事実の不確かさには検証、といったように、問題の性質に合った人間の介入が、AIのパフォーマンスを効果的に向上させます。人間応答アプリは、これらの多様な応答形式を柔軟にサポートできるインターフェースを備える必要があります。

4. 人間参加型AI（Human-in-the-Loop）システムの事例調査

提案されている人間応答アプリは、「人間参加型AI（Human-in-the-Loop AI、HITL AI）」の一形態と捉えることができます。HITLは、AIシステムのライフサイクル（学習、評価、運用）に人間の知性や判断を組み込むアプローチであり、AI単独では達成困難な精度、信頼性、適応性を実現することを目的としています [27]。既存のHITLシステムの事例を調査することで、人間応答アプリの設計に役立つ知見を得ることができます。

4.1. HITLの一般的な仕組みと目的

HITLシステムでは、人間は主に以下の役割を果たします [46]:

- データのアノテーション: AIモデルの学習に必要なラベル付きデータを作成する（例：画像内のオブジェクトにタグ付けする）[46]。
- モデルの評価・検証: AIモデルの出力（予測、分類など）を人間が評価し、その精度や妥当性を検証する [46]。
- フィードバックと修正: AIが低い信頼度で予測した場合や、誤った判断をした場合に、人間が介入して修正し、その結果をフィードバックすることでモデルを継続的に改善する [45]。
- 意思決定: AIが生成した推奨事項や分析結果に基づいて、最終的な意思決定を人間が行う [45]。

HITLの導入により、AIシステムの精度向上、バイアスの低減、透明性の向上、予期せぬ状況への対応力強化、ユーザー信頼の醸成といった利点が得られます [27]。一方で、コスト、人間の可用性への依存、ヒューマンエラーの可能性、スケーラビリティの課題なども存在します [27]。

4.2. HITLの具体的な応用事例

HITLは様々な分野で活用されています。

- コンテンツモデレーション: SNSやオンラインプラットフォームにおいて、AIが不適切コンテンツの候補を検出し、人間が最終的な判断を下す。人間は文脈やニュアンスを理解できるため、AIだけでは難しい判断が可能です [51]。
- 医療診断支援: AIが医療画像（レントゲン写真など）を分析し、異常の可能性を指摘、医師がその結果を確認・検証して最終診断を下す [46]。
- 自動運転: 自動運転車は基本的にAIで制御されますが、予期せぬ状況や複雑な交通環境では人間の介入が必要になる場合があります [21]。また、人間の運転データを収集・分析することでAIモデルを改善します [51]。
- 顧客サービス: チャットボットが一次対応を行い、複雑な問い合わせや感情的な対応が必要な場合に人間のオペレーターに引き継ぐ [46]。
- データ処理（**OCR**、請求書処理など）: OCR（光学的文字認識）ソフトウェアが文書からテキストを抽出する際、AIの信頼度が低い箇所（例：手書き文字、不鮮明な印字）を人間が確認・修正する [50]。これにより、財務やヘルスケアなど、精度が極めて重要な分野でのエラーを最小限に抑えます [50]。同様に、請求書処理やIDカード検証、データ匿名化などでも活用されています [50]。
- 検索エンジン: ユーザーが検索結果からどのサイトをクリックするかを観察することで、検索アルゴリズムの精度を向上させます [51]。
- **CAPTCHA:** 人間には容易だが機械には困難なタスク（歪んだ文字の認識、画像内の特定オブジェクトの選択）をユーザーに課すことで、ボットと人間を区別します。このプロセスで収集されたデータが、間接的にAIの学習データとして利用されることもあります [51]。

4.3. HITLシステムの設計パターン

HITLシステムの設計においては、人間の関与の仕方によっていくつかのパターンが存在します。

- ループの開始点 **(Human at the Beginning):** 主にAIモデルの初期学習段階で、人間が大量のデータにラベル付けを行うケースです [50]。
- ループの途中 **(Human in the Loop):** AIの処理プロセス中に、特定の判断や検証、修正のために人間が介入するケース。提案されている人間応答アプリはこのパターンに該当します。AIが低い信頼度を示した場合や、特定のトリガー（倫理的チェックなど）が発生した場合に人間が呼び出されます [45]。
- ループの終点 **(Human at the End):** AIが処理を完了した後、最終的な承認やレビューを人間が行うケースです [50]。

LangGraphのようなフレームワークでは、グラフの実行を特定のノードで一時停止させ、人間の入力を受け付けてから再開する「interrupt」機能を提供しており、承認/拒否、状態編集、追加情報収集といったHITLワークフローの実装を支援します [45]。また、Buildtime HITL（開発時に人間の専門知識をエージェントの思考プロセス設計に組み込む）とRuntime HITL（実行時に人間が介入する）という分類も存在します [56]。

表1：**HITL**（人間参加型**AI**）の相互作用パターン

パターン名	主な目的	人間の役割	**AI**との連携タイミング	代表的な例
データアノテーション	AIモデルの初期学習データ作成	データへのラベル付け、タグ付け、境界線描画など	モデル学習前 (Beginning of the Loop)	画像認識用データセット作成、テキスト分類用データ作成 [46]
モデル検証/評価	AIモデルの出力精度・妥当性の確認	AIの予測/分類結果の正誤判定、品質評価	モデル学習後、または運用中 (End of the Loop)	医療診断支援AIの医師によるレビュー [46]、コンテンツ推薦の評価
リアルタイム修正/フィードバック	AIの低信頼度予測やエラーの即時修正、継続的改善	AIの出力結果の修正、代替案の提示、理由のフィードバック	AI運用中、AIが支援を要求した際 (In the Loop)	OCR誤認識の修正 [50]、チャットボットのエスカレーション [46]、本提案アプリ

意思決定/承認	AIの提案に基づく最終判断、リスク管理	AIの推奨アクションの承認/拒否、パラメータ調整、実行可否判断	AI運用中、特定の判断ポイント (In/End of the Loop)	自動取引システムのトレーダーによる承認、AIによるインシデント対応計画の承認 [45]
能動学習 **(Active Learning)**	効率的なモデル改善	AIが選択した「学習効果の高い」データへのラベル付け	モデル学習/運用中 (In the Loop)	ラベル付けコストが高い場合の効率的なモデル訓練 [47]
強化学習 **(Reinforcement Learning)**	試行錯誤によるAIの行動学習	AIの行動に対する報酬/ペナルティの付与（フィードバック）	モデル学習/運用中 (In the Loop)	ゲームAIの訓練、ロボット制御 [47]
対話型機械学習 **(Interactive ML)**	人間との対話を通じたシステム構築/改善	システムへの指示、デモンストレーション、パラメータ調整、継続的なフィードバック	システム設計・開発・運用全般 (Human-AI Interaction)	音楽生成ツールの操作 [49]、エージェントとの協調作業 [56]

これらの事例と設計パターンは、人間応答アプリがAIのどの「悩み」に対して、どのタイミングで、どのような形式の介入を行うべきかを設計する上で重要な示唆を与えます。特に、AIがリアルタイムで支援を要求し、人間が修正や判断を提供してプロセスを続行させる「ループの途中 (In the Loop)」のパターンが、本アプリケーションのコアとなるでしょう。

5. アプリケーションの基本機能と技術的実現性

これまでの調査に基づき、生成AIの「悩み」に対応する人間応答アプリに必要な基本機能、AIモデルとの連携方法、そして開発に適した技術スタックについて考察します。

5.1. アプリケーションに必要な基本機能

人間応答アプリが効果的に機能するためには、以下の基本機能を備える必要があります。

- **AIからのリクエスト表示:** AIが「悩んでいる」状況（曖昧な指示、倫理的判断、事実確認など）と、AIからの具体的な質問や支援要求を、人間が理解できる形式で明確に表示する機能。AIの内部状態や文脈情報を適切に提示し、人間が判断を下すための十分な情報を提供することが重要です [49]。
- 人間用入力インターフェース: 人間が応答を入力するためのインターフェース。これには、状況に応じて以下のような形式をサポートする必要があります。
    - 選択肢からの選択（例：解釈の選択、承認/拒否）[45]

    - 自由記述テキスト入力（例：指示の明確化、アイデア提供、修正案）[16]
    - 評価・スコアリング（例：生成物の品質評価）[51]
    - ファイルアップロード（例：参照すべき資料の提供）
    - 構造化データ入力（例：パラメータ調整）[56] 直感的で効率的なUI/UXデザインが求められます [52]。
- 応答の**AI**への送信**:** 人間が入力した応答を、AIモデルが解釈・利用できる形式で、連携しているAIシステムへ送信する機能。
- 履歴管理**:** AIからのリクエスト、人間の応答、そしてその結果（AIが応答をどう利用したか）の履歴を記録・管理する機能。これにより、過去の事例を参照したり、人間の応答品質を評価したり、監査証跡として利用したりすることが可能になります [32]。
- ユーザー管理と認証**:** 応答を提供する人間のユーザーアカウントを管理し、安全に認証する機能 [53]。必要に応じて、応答者の専門性に基づいた役割ベースのアクセス制御（RBAC）も考慮します [57]。
- 通知機能**:** AIから新たな支援要求があった際に、担当する人間に通知する機能（例：プッシュ通知、メール）[58]。

5.2. AIモデルとの連携アーキテクチャ

人間応答アプリを生成AIモデルと連携させる方法は、主に以下の選択肢が考えられます。

- **API**連携**:**
    - **AI**側**API**利用**:** 人間応答アプリが、AIモデル提供者（OpenAI, Anthropic, Googleなど）の提供するAPIを利用して、AIの状態を取得したり、人間の応答をAIに送信したりする方法。AIモデル側が外部からの介入を受け付けるインターフェースを持っている必要があります。
    - 応答アプリ側**API**提供**:** 人間応答アプリがAPIエンドポイントを提供し、AIシステム側（あるいはAIを制御するオーケストレーション層）が、支援が必要な場合にこのAPIを呼び出す方法。AIシステム側で、人間への問い合わせが必要な状況を検知し、APIを呼び出すロジックを実装する必要があります。これは、AIエージェントフレームワーク（LangChain, LangGraphなど）の「ツール」や「カスタム関数」として実装されることが多いでしょう [45]。
- メッセージキュー連携**:** AIシステムと人間応答アプリが、非同期メッセージキュー（例：RabbitMQ, Kafka, AWS SQS, Google Pub/Sub）を介してリクエストと応答を送受信する方法。疎結合な連携が可能で、スケーラビリティや耐障害性に優れていますが、リアルタイム性が求められる場合にはレイテンシの考慮が必要です。
- **Model Context Protocol (MCP)** 連携**:**
    - AIシステムがMCPクライアントとして動作し、人間応答アプリがMCPサーバーとして機能を提供する方法。人間応答アプリは、「応答要求」を受け付けるツールや、関連情報を提供するリソースをMCP経由で公開します [3]。
    - このアプローチは、AIシステム側がMCPに対応している場合に有効です。MCPは標

準化されたプロトコルであるため、一度MCPサーバーを構築すれば、異なるMCP対応AIクライアント（例：Claude Desktop, Cursor IDEなど [3]）から利用できる可能性があります。
    - MCPは、クライアント（AIアプリ）とサーバー（ツール提供者）間の通信を標準化し [2]、stdio（ローカルプロセス間通信）やHTTP/SSE（リモート通信）といったトランスポート層をサポートします [11]。
    - MCPサーバーの実装例は、Python [7]、C# [8]、Java (LangChain4j経由) [60]、Node.js (npmパッケージ) [62] など、様々な言語で提供・紹介されています。GitHub MCPサーバー [60] やファイルシステムサーバー [8] など、具体的なツール連携の例もあります。
    - ただし、MCP自体にもセキュリティリスク（後述）が存在するため、導入には注意が必要です。

連携方式の選択は、対象とするAIシステムのアーキテクチャ、リアルタイム性の要求度、開発リソース、そしてMCPのような標準プロトコルへの準拠方針によって決定されるべきです。特に、人間応答アプリはAIの処理を一時的にブロックする可能性があるため、応答遅延がAIシステム全体のパフォーマンスに与える影響を最小限に抑えるアーキテクチャ設計が重要です。非同期処理を基本としつつ、人間からの応答を待つ間のAI側のタイムアウト処理や、応答がない場合の代替処理フローなどを考慮する必要があります [58]。

5.3. 技術スタックの候補

リアルタイム性、AI連携、Webアプリケーションとしての要件を考慮すると、以下のような技術スタックが候補として挙げられます。

- プログラミング言語:
    - **Python:** AI/ML分野で最もエコシステムが充実しており、関連ライブラリ（TensorFlow, PyTorch, LangChain, Transformersなど）が豊富。Webフレームワーク（Flask, Django, FastAPI）も成熟しています [64]。FastAPIは非同期処理に強く、API開発に適しています [65]。MCPのPython SDKも提供されています [61]。
    - **Node.js (JavaScript/TypeScript):** 非同期I/O処理に優れ、リアルタイムWebアプリケーション（WebSocket利用など）に適しています。フルスタックJavaScript開発が可能 [65]。TensorFlow.jsを使えばブラウザ上でのML処理も可能です [65]。MCPサーバーのnpmパッケージも存在します [62]。Next.jsはモダンなフルスタック開発フレームワークとして人気があります [65]。
    - **Java:** エンタープライズシステムで広く利用されており、堅牢性やスケーラビリティに優れています。Spring Bootフレームワークが一般的 [65]。Deep Java Library (DJL) などAIライブラリも存在します [65]。LangChain4jはJava向けのLangChain実装で、MCPクライアント機能も提供します [60]。
    - **C#:**.NET環境での開発に適しており、ASP.NET Coreは高性能なWebフレームワークです。MicrosoftがMCP C# SDKを提供しており、.NET環境でのMCPサーバー/クライアント開発が容易になっています [8]。

- **Web**フレームワーク**:** 上記言語に対応する主要フレームワーク（FastAPI, Django, Express, NestJS, Spring Boot, ASP.NET Coreなど）。リアルタイム通信のためにWebSocketをサポートするものが望ましいです。
- フロントエンド**:** React, Vue.js, AngularなどのモダンJavaScriptフレームワーク。ユーザーインターフェースの構築とインタラクティブ性の実現に利用します。Next.js [65] やNuxt.js のようなフルスタックフレームワークも選択肢です。
- データベース**:**
  - リレーショナル**DB (PostgreSQL, MySQL):** 構造化されたデータ（ユーザー情報、リクエスト履歴など）の管理に適しています [65]。PostgreSQLは拡張性に優れ、ベクトルデータ（pgvector）なども扱えるためAI用途で人気があります。Neonはサーバーレス Postgresを提供します [65]。
  - **NoSQL DB (MongoDB, Cassandra):** 柔軟なスキーマを持ち、大量の非構造化/半構造化データ（会話ログなど）の扱いに適しています [67]。MongoDBはJavaScriptスタックでよく利用されます [65]。
- **AI/ML**連携**:** 利用するAIモデルのAPIに対応したSDK（OpenAI Python Library, Anthropic SDKなど）。複雑なAIワークフローを構築する場合は、LangChain [59]、LangGraph [45]、AutoGen [63] などのフレームワークの利用を検討します。MCP連携を行う場合は、対応するSDK（Python [61], C# [8], Java [60] など）を利用します。
- インフラストラクチャ**/**デプロイメント**:**
  - クラウドプラットフォーム **(AWS, Azure, GCP):** マネージドサービス（データベース、コンピューティング、AIサービス、メッセージキューなど）が豊富で、スケーラビリティと運用効率に優れます [64]。
  - コンテナ化 **(Docker, Kubernetes):** アプリケーションのパッケージ化とデプロイ、スケーリングを容易にします [64]。KubeflowはKubernetes上でのMLワークフロー管理を支援します [65]。
  - サーバーレス**:** AWS Lambda, Google Cloud Functions, Azure Functionsなどのサーバーレスコンピューティングは、イベント駆動型の処理やAPIバックエンドに適しており、スケーラビリティとコスト効率に貢献します。
  - **MLOps**ツール**:** モデルのライフサイクル管理が必要な場合、MLflow, Kubeflow, DVCなどのツールが役立ちます [65]。

表**2**：人間応答**AI**アプリケーション向け技術スタック比較

スタックタイプ	主要コンポーネント例	主な利点	主な欠点	最適なケース**/**チーム
Python中心	Python, FastAPI/Django,	強力なAIエコシステム、豊富なライ	JavaScript系と比較してフロントエン	AI/ML処理がコア、データ分析多

	PostgreSQL/MySQL, React/Vue, LangChain/PyTorch/TensorFlow	ブラリ、データサイエンスとの親和性 [65]	ド開発の複雑さ、GILによる並行処理の制約（非同期で緩和可）	用、Pythonに慣れたチーム [65]
Node.js中心 (JavaScript/TS)	Node.js, Express/NestJS, MongoDB/PostgreSQL, React/Vue/Angular, TensorFlow.js	高いリアルタイム性能（非同期I/O）、フルスタックJS開発、NPMエコシステム、ブラウザML [65]	Pythonに比べAI/MLライブラリの選択肢が限定的、CPU負荷の高い計算処理には不向きな場合あり	リアルタイム性が最重要、Web開発中心、JavaScript/TypeScriptに慣れたチーム [65]
Java中心	Java, Spring Boot, PostgreSQL/MySQL, Angular/React, DJL/LangChain4j	エンタープライズレベルの堅牢性・安定性、強力な型システム、大規模開発向け [65]	Python/JSに比べ開発速度が遅くなる可能性、AI/MLエコシステムの成熟度は劣る	既存Javaシステムとの連携、大規模・高信頼性要求、Javaに慣れたエンタープライズチーム [65]
C#/.NET中心	C#, ASP.NET Core, PostgreSQL/SQL Server, Blazor/React, ML.NET/MCP C# SDK	高性能Webフレームワーク、Microsoftエコシステムとの親和性、MCP SDKの利用容易性 [8]	AI/MLエコシステムはPythonに劣る、Windows環境以外での利用経験が浅い場合あり	.NET環境での開発、Microsoft技術スタック利用、MCP連携をC#で行いたいチーム
クラウドネイティブ/サーバーレス	AWS Lambda/GCP Functions/Azure Functions, API Gateway, DynamoDB/Firestore, SQS/PubSub	高いスケーラビリティ、運用負荷軽減、従量課金によるコスト効率 [65]	ベンダーロックインのリスク、ローカルでのテスト/デバッグの複雑さ、コールドスタート問題	スケーラビリティ重視、運用負荷を最小化したい、イベント駆動型アーキテクチャに適したユースケース [65]

技術スタックの選択は、プロジェクトの具体的な要件（特にリアルタイム性）、チームの既存スキル、開発期間、予算、そして将来的な拡張性などを総合的に考慮して決定する必要があります [65]。

6. プライバシーおよびセキュリティ上の考慮事項

人間応答アプリは、AIとの対話データや人間からの入力データを扱うため、プライバシーとセ

キュリティに対する配慮が極めて重要です。これらのデータには機密情報や個人情報が含まれる可能性があり、不適切な取り扱いは深刻なリスクにつながります [22]。

6.1. データプライバシーに関する考慮事項

- 扱うデータの機密性: AIが悩んでいる状況のコンテキスト（元のプロンプト、AIの途中生成物など）や、それに対する人間の応答（修正案、判断理由など）は、個人情報（PII）、企業の機密情報、著作権に関わる情報など、センシティブな内容を含む可能性があります [22]。
- 法的コンプライアンス: GDPR（EU一般データ保護規則）、CCPA（カリフォルニア州消費者プライバシー法）、HIPAA（医療保険の相互運用性と説明責任に関する法律、医療情報の場合）など、適用されるデータ保護法規制を遵守する必要があります [26]。これには、データ収集・利用目的の明確化、同意取得、データ主体の権利（アクセス権、訂正権、削除権など）の保証が含まれます。
- プライバシー保護技術:
  - データ最小化: アプリケーションの機能に必要な最小限のデータのみを収集・保持します [32]。
  - 匿名化・仮名化: 可能であれば、個人を特定できないようにデータを匿名化または仮名化して処理します [31]。
  - 暗号化: 保存データ（at rest）および通信中のデータ（in transit）を強力な暗号化で保護します [31]。
  - アクセス制御: データへのアクセス権限を厳格に管理し、必要最小限の担当者のみがアクセスできるようにします（RBAC）[32]。
  - 透明性: ユーザー（AIシステムの利用者および応答を提供する人間）に対して、どのようなデータが収集・利用され、どのように保護されているかを明確に説明します [23]。
  - データ保持期間: 法令やポリシーに基づき、データの保持期間を定め、期間終了後は安全に削除します [52]。
  - 定期的な監査: データプライバシーに関するポリシーと実践状況を定期的に監査します [26]。

6.2. システムコンポーネントと通信のセキュリティ対策

- 認証と認可: 人間応答アプリにアクセスするユーザーを安全に認証します（例：多要素認証）。API連携を行う場合は、APIキー管理やOAuthなどを用いて、システム間の通信を保護します [58]。応答者の役割に応じてアクセス権限を制御します（RBAC）[57]。
- インフラストラクチャセキュリティ: アプリケーションをホストする環境（クラウド、オンプレミス）を保護します。ファイアウォール設定、侵入検知/防御システム（IDS/IPS）、脆弱性管理、定期的なセキュリティスキャンなどを実施します [31]。データベースやストレージへのアクセスも適切に保護します。
- セキュアコーディング: OWASP Top 10などに挙げられる一般的なWebアプリケーションの脆弱性（クロスサイトスクリプティング(XSS)、SQLインジェクションなど）を防ぐためのセキュアコーディングプラクティスに従います。

- 依存関係の管理: 使用するライブラリやフレームワークの脆弱性を定期的にチェックし、パッチを適用します [69]。
- ロギングとモニタリング: システムの動作ログ、アクセスログ、エラーログなどを収集・監視し、不正アクセスや異常な挙動を早期に検知できる体制を構築します [31]。
- インシデント対応計画: セキュリティインシデントが発生した場合の対応計画を策定し、定期的に訓練を実施します [31]。

6.3. HITLおよびAIシステム特有のリスク

- ヒューマンエラーと内部脅威: 人間応答者は、意図せずに誤った情報を提供したり、操作ミスをしたりする可能性があります [27]。悪意を持った内部者が情報を漏洩したり、システムを悪用したりするリスクも存在します。対策として、明確なガイドラインの提供、トレーニング、作業内容の監査、ダブルチェック体制などが考えられます [22]。
- バイアスの混入・増幅: 応答を提供する人間の持つバイアスが、フィードバックを通じてAIモデルに学習され、バイアスが固定化・増幅されるリスクがあります [26]。多様なバックグラウンドを持つ応答者チームの編成や、バイアスを検知・軽減する仕組みの導入が必要です [32]。
- プロンプトインジェクション: AIへの指示（プロンプト）に悪意のある命令を埋め込む攻撃です [29]。人間応答アプリがAIからのコンテキストを表示する際や、人間からの応答をAIに渡す際に、この攻撃経路が生まれる可能性があります。入力（AIからのコンテキスト、人間からの応答）に対する厳格なサニタイズと検証、そして応答者への注意喚起が必要です [31]。
- データ漏洩リスク: 機密性の高いコンテキスト情報が、応答を提供するために人間に表示される過程で漏洩するリスクがあります。表示する情報の範囲を最小限に留める、マスキング処理を行うなどの対策が必要です [67]。
- 責任の所在の曖昧さ: 人間とAIが協調してタスクを実行するシステムでは、エラーや問題が発生した場合に、その責任が人間にあるのかAIにあるのか、あるいはシステム設計にあるのかを特定することが困難になる場合があります [21]。詳細なログ記録と、各ステップにおける役割と責任の明確化が重要です [32]。

6.4. MCP連携における固有のセキュリティリスク

もしAIシステムとの連携にMCPを採用する場合、MCP特有のセキュリティリスクにも注意が必要です。

- コンテキストポイズニング: 攻撃者がMCPサーバーが参照するデータソース（ファイル、DBなど）を改ざんし、悪意のある情報やプロンプトをAIに注入する [69]。
- MCPサーバーの侵害: MCPサーバー自体が侵害されると、そのサーバーが保持する連携先サービスへの認証情報（APIキー、OAuthトークンなど）が窃取され、広範囲な不正アクセスにつながる可能性があります ("Keys to the kingdom" シナリオ) [69]。
- 認証・認可の不備: MCPサーバーの実装によっては、リクエスト元のクライアント（AIアプ

リ）の認証や、要求された操作に対する認可が適切に行われない場合があります [69]。
- トークン窃取: MCPサーバーが連携サービスのために保持するOAuthトークンなどが攻撃対象となります [71]。
- 過剰な権限: MCPサーバーが連携先サービスに対して必要以上に広範な権限（例：メールの読み取りだけでなく送受信権限）を要求・保持している場合、侵害時の被害が拡大します [70]。最小権限の原則を適用する必要があります [70]。
- サプライチェーンリスク: 脆弱性を含むオープンソースのMCP実装や、信頼できないサードパーティ製のMCPサーバーを利用することによるリスク [69]。利用する実装の十分な検証が必要です [69]。
- ツールシャドウイング/ラグプル: 悪意のあるMCPサーバーが、信頼できる別のMCPサーバーのツール定義を上書きしたり、後から悪意のある機能に変更したりする [72]。

これらのリスクへの対策としては、MCPサーバーへのアクセス制御の強化、入力の検証・サニタイズ、最小権限の原則の適用、信頼できる実装の利用、通信の暗号化、そして重要な操作に対する人間による最終確認プロンプトの導入などが挙げられます [11]。

表3：人間応答AIシステムにおけるセキュリティ・プライバシーリスクと対策

リスク領域	説明	潜在的影響	対策（技術的・手続き的）
データプライバシー侵害	機密情報（個人情報、企業秘密など）を含むAIコンテキストや人間応答の不適切な収集、利用、保管、漏洩 [26]	法令違反（GDPR等）、罰金、信用の失墜、ユーザー被害	データ最小化、匿名化/仮名化、暗号化、厳格なアクセス制御(RBAC)、透明性の確保、同意取得、データ保持ポリシー策定・遵守、定期監査 [31]
プロンプトインジェクション	AIへの指示や人間応答に悪意のある命令を埋め込み、AIに意図しない操作（情報漏洩、不正実行など）を行わせる攻撃 [29]	情報漏洩、不正なシステム操作、サービス妨害	入力（AIコンテキスト、人間応答）の厳格なサニタイズ・検証、出力エンコーディング、コンテキスト分離、応答者への注意喚起/トレーニング、重要な操作前の人間による確認 [31]
不正アクセス/権限昇格	認証・認可の不備を突かれ、不正ユーザーがシステムにアクセスした	情報漏洩、データ改ざん、システム停止	強力な認証（MFA）、セキュアなAPIキー/トークン管理、最小権限の原

	り、必要以上の権限で操作したりする		則に基づくアクセス制御（RBAC）、セキュアなインフラ設定、脆弱性管理 [32]
バイアスの混入/増幅	人間応答者のバイアスがフィードバックを通じてAIに学習され、差別的な結果や不公平な判断を助長する [26]	差別、不公平な扱いの助長、社会的信用の失墜	多様な応答者チームの編成、バイアス検出ツールの導入、明確な評価基準の設定、定期的なバイアス監査、応答者トレーニング [32]
ヒューマンエラー	応答者が意図せず誤った情報を提供したり、操作ミスをしたりする [27]	AIの誤学習、不正確なアウトプット、プロセスの遅延	明確な指示とガイドライン、直感的なUI/UX、入力内容の確認ステップ、ダブルチェック体制、トレーニング、作業ログの監査 [31]
責任の所在の曖昧さ	エラー発生時に、人間、AI、システムのいずれに責任があるかの特定が困難 [21]	問題解決の遅延、再発防止策の不備、法的・倫理的問題	詳細なログ記録（誰が、いつ、何を、なぜ）、明確な役割分担と責任範囲の定義、透明性の高いプロセス設計 [32]
MCPサーバー侵害 **(MCP**利用時**)**	MCPサーバーが侵害され、保持する認証情報が窃取される [69]	連携先サービスへの広範な不正アクセス、情報漏洩、不正操作	サーバーのセキュリティ強化、認証情報の安全な保管（シークレット管理）、最小権限での連携、アクセス監視、脆弱性スキャン [69]
コンテキストポイズニング **(MCP**利用時**)**	攻撃者がMCPサーバーのデータソースを改ざんし、悪意のある情報をAIに注入する [69]	AIの誤誘導、情報漏洩、不正なツール実行	データソースのアクセス制御と整合性チェック、入力コンテキストの検証・サニタイズ、信頼できないソースからの分離 [69]

効果的なHITLシステムを構築するには、人間が介入しやすいように透明性を高めることと、その透明性が新たなセキュリティリスクを生まないようにすることのバランスを取る必要があります。人間応答者に十分なコンテキストを提供しつつ、機密情報への不必要なアクセスを防ぎ、表示される情報が悪用されないように保護する設計が求められます [23]。データのマスキング、

情報の要約、明確で安全な説明生成などが有効なアプローチとなり得ます [52]。

7. 結論と戦略的推奨事項

7.1. 調査結果の要約

本調査により、生成AIが「悩む」多様な状況（曖昧さ、倫理、創造性、事実性など）が存在し、それに対して人間が適切な応答を提供することでAIの能力を補完・向上させるアプリケーション（人間応答アプリ）の潜在的な価値が確認されました。ユーザーが言及した「MCP」は、AIと外部ツールを連携させるための標準プロトコル「Model Context Protocol」を指す可能性が高いものの、本アプリの実現方法はMCP連携に限定されず、API連携やメッセージキュー連携など複数の選択肢が存在します。効果的な人間応答の形式はAIの「悩み」の種類に応じて異なり、既存のHITLシステムの事例はアプリ設計の指針となります。アプリケーションには、リクエスト表示、多様な入力形式、応答送信、履歴管理などの基本機能が求められ、技術スタックの選定においてはリアルタイム性、AI連携の容易さ、チームの専門性が重要な考慮事項となります。そして何より、機密情報を扱う可能性があるため、プライバシー保護とセキュリティ対策を設計段階から組み込むことが不可欠です。

7.2. アプリケーションの実現可能性

技術的には、本報告書で概説した機能を持つ人間応答アプリの開発は十分に実現可能です。主要な課題は以下の点に集約されると考えられます。

- リアルタイム性と**UX:** AIの処理を妨げずに、人間からの応答をタイムリーに収集し、AIにフィードバックする低遅延な連携メカニズムと、人間応答者にとって効率的で負担の少ないユーザーインターフェースの設計。
- コンテキストの適切な提示**:** 人間が的確な判断を下せるだけの十分なコンテキストを提供しつつ、機密情報漏洩やセキュリティリスクを回避するバランス。
- 多様な「悩み」への対応**:** AIが示す多様な「悩み」の種類と、それに応じた適切な応答形式をどのようにハンドリングし、人間に提示するか。
- セキュリティとプライバシー**:** 設計・実装・運用の各段階における徹底したセキュリティ対策とプライバシー保護の実践。

7.3. 戦略的推奨事項

人間応答アプリの開発を成功させるために、以下の戦略的アプローチを推奨します。

1. スモールスタートと段階的拡張**:** まず、特定のAIモデルやユースケースにおける、1〜2種類の明確な「悩み」の状況（例：特定の種類の曖昧さの解消、特定の情報のファクトチェック）に焦点を当てて開発を開始します。成功体験を積み重ねながら、対応範囲を段階的に広げていくアプローチが現実的です。
2. ユーザーエクスペリエンス（**UX**）の最優先**:** 応答を提供する人間が、迅速かつ正確に判断・応答できるような、直感的で効率的なインターフェース設計に重点を置きます [32]。必要

なコンテキストを過不足なく、分かりやすく提示することが鍵となります。

3. 反復的な開発とフィードバック: アジャイルな開発プロセスを採用し、早期にプロトタイプを作成して、実際に応答を提供するユーザーからのフィードバックを収集・反映させながら、継続的に改善を行います [49]。シンプルなHITLメカニズム（例：単純な承認/拒否）から始め、必要に応じてより複雑な対話機能へと進化させることも検討します。
4. セキュリティ・プライバシー・バイ・デザイン: 開発の初期段階からセキュリティとプライバシーの要件を定義し、設計に組み込みます [31]。潜在的なリスク（表3参照）を特定し、それに対する技術的・手続き的な対策を計画・実装します。
5. 技術スタックの慎重な選定: チームの技術的専門知識、求められるリアルタイム性能、連携対象のAIシステムとの親和性、スケーラビリティ要件などを考慮して、最適な技術スタックを選択します（表2参照）[65]。PythonやNode.jsは有力な候補ですが、既存環境との整合性も重要です。
6. 連携方式の評価: 対象とするAIシステムやオーケストレーション層のアーキテクチャを考慮し、API連携、メッセージキュー連携、あるいはMCP連携の中から、要件に最も適した連携方式を選択・設計します。
7. 人間応答者の役割定義: 誰が応答を提供するのか（専門家、一般ユーザーなど）、どのようなスキルや知識が必要か、リクエストの割り当てや処理フローをどのように管理するかを明確に定義します [50]。
8. 継続的な監視と適応: アプリケーションの稼働後も、システムのパフォーマンス、人間応答の品質と効率、AI側のエラー発生傾向などを継続的に監視し、必要に応じてシステムや運用プロセスを改善・適応させていきます [31]。AI技術や関連プロトコルの進化にも注意を払います。

7.4. 将来展望

AI技術は急速に進化していますが、完全な自律性にはまだ課題が多く、特に複雑な判断、倫理的な配慮、創造性、高度な事実性が求められる場面では、人間の関与が引き続き重要となります [52]。提案されている人間応答アプリのようなHITLシステムは、AIの限界を補い、その能力を最大限に引き出すための鍵となるでしょう。将来的には、このようなシステムを通じて収集された人間による高品質なフィードバックデータが、次世代AIモデルの学習と改善に貢献することも期待されます [56]。AIと人間が効果的に協調する未来において、本アプリケーションのようなインターフェースは、ますますその重要性を増していくと考えられます。

引用文献

1. [2501.04040] A Survey on Large Language Models with some Insights on their Capabilities and Limitations - arXiv, 4月 22, 2025にアクセス、https://arxiv.org/abs/2501.04040
2. Understanding the Model Context Protocol | Frontegg, 4月 22, 2025にアクセス、https://frontegg.com/blog/model-context-protocol
3. Model Context Protocol (MCP) Explained - Humanloop, 4月 22, 2025にアクセス、

https://humanloop.com/blog/mcp
4. techcommunity.microsoft.com, 4月 22, 2025にアクセス、https://techcommunity.microsoft.com/blog/educatordeveloperblog/unleashing-the-power-of-model-context-protocol-mcp-a-game-changer-in-ai-integrat/4397564#:~:text=MCP%20is%20a%20protocol%20designed.beyond%20their%20built%2Din%20knowledge.
5. What is MCP (Model Context Protocol)? | Zapier, 4月 22, 2025にアクセス、https://zapier.com/blog/mcp/
6. ちゃんと理解したい初心者のための「MCP」まとめ #生成AI - Qiita, 4月 22, 2025にアクセス、https://qiita.com/to3izo/items/99dd3cde237c2e5a007f
7. Model Context Protocol (MCP): A Guide With Demo Project - DataCamp, 4月 22, 2025にアクセス、https://www.datacamp.com/tutorial/mcp-model-context-protocol
8. Build a Model Context Protocol (MCP) server in C# - .NET Blog, 4月 22, 2025にアクセス、https://devblogs.microsoft.com/dotnet/build-a-model-context-protocol-mcp-server-in-csharp/
9. A beginners Guide on Model Context Protocol (MCP) - OpenCV, 4月 22, 2025にアクセス、https://opencv.org/blog/model-context-protocol/
10. Model Context Protocol (MCP): Integrating Azure OpenAI for Enhanced Tool Integration and Prompting | Microsoft Community Hub, 4月 22, 2025にアクセス、https://techcommunity.microsoft.com/blog/azure-ai-services-blog/model-context-protocol-mcp-integrating-azure-openai-for-enhanced-tool-integratio/4393788
11. model-context-protocol-resources/guides/mcp-server-development-guide.md at main - GitHub, 4月 22, 2025にアクセス、https://github.com/cyanheads/model-context-protocol-resources/blob/main/guides/mcp-server-development-guide.md
12. Model Context Protocol (MCP): A comprehensive introduction for developers - Stytch, 4月 22, 2025にアクセス、https://stytch.com/blog/model-context-protocol-introduction/
13. 【超初心者向け】MCPって何なん？ どう使うん？ #ChatGPT - Qiita, 4月 22, 2025にアクセス、https://qiita.com/benjuwan/items/0fb8cc0f034f8b0d942f
14. Core architecture - Model Context Protocol, 4月 22, 2025にアクセス、https://modelcontextprotocol.io/docs/concepts/architecture
15. AIをもっと賢くする魔法のルール！「MCP」ってなんだろう？徹底解説 - note, 4月 22, 2025にアクセス、https://note.com/redcord/n/n3dd127ed6012
16. Expert's Guide: Generative AI Prompts for Maximum Efficiency - HatchWorks, 4月 22, 2025にアクセス、https://hatchworks.com/blog/gen-ai/generative-ai-prompt-guide/
17. (PDF) Tackling the Ambiguity Challenge with Generative Artificial Intelligence: AICMA, A Framework for Identification, Classification and Mitigation of Ambiguity - ResearchGate, 4月 22, 2025にアクセス、https://www.researchgate.net/publication/390460768_Tackling_the_Ambiguity_Challenge_with_Generative_Artificial_Intelligence_AICMA_A_Framework_for_Identification_Classification_and_Mitigation_of_Ambiguity

18. OVERCOMING THE AMBIGUITY REQUIREMENT USING GENERATIVE AI - DiVA portal, 4月 22, 2025にアクセス、https://www.diva-portal.org/smash/get/diva2:1931301/FULLTEXT01.pdf
19. ④生成AIが苦手なことを知ってますか？改善方法と使い方のヒント - note, 4月 22, 2025にアクセス、https://note.com/sakura_digilab/n/nacbc23749c75
20. 生成AIが引き起こすハルシネーションとは？嘘を防ぐための対策プロントを紹介 | マナビタイム, 4月 22, 2025にアクセス、https://manab-it.com/magazine/category/useful/ai/139
21. 9 AI Dilemmas That Challenge Generative AI Ethics - Creatopy, 4月 22, 2025にアクセス、https://www.creatopy.com/blog/ethics-of-generative-ai/
22. The Ethical Dilemma in Generative AI: Survey Results from 500 Businesses - ScaleupAlly, 4月 22, 2025にアクセス、https://scaleupally.io/blog/ai-ethical-dilemma/
23. The ethical dilemmas of AI | USC Annenberg School for Communication and Journalism, 4月 22, 2025にアクセス、https://annenberg.usc.edu/research/center-public-relations/usc-annenberg-relevance-report/ethical-dilemmas-ai
24. オリンピックで再燃したAIの倫理問題を考える。最新の取り組み事例も紹介 - AIsmiley, 4月 22, 2025にアクセス、https://aismiley.co.jp/ai_news/what-is-the-ethical-issue-of-ai-that-has-become-more-important-due-to-the-evolution-of-technology/
25. AIと倫理 – AIは倫理をどう見る？これまでの事例や取り組みを紹介 - AINOW, 4月 22, 2025にアクセス、https://ainow.ai/2020/02/20/182887/
26. The growing data privacy concerns with AI: What you need to know - DataGuard, 4月 22, 2025にアクセス、https://www.dataguard.com/blog/growing-data-privacy-concerns-ai/
27. The Role of Human-in-the-Loop: Navigating the Landscape of AI Systems, 4月 22, 2025にアクセス、https://humansintheloop.org/the-role-of-human-in-the-loop-navigating-the-landscape-of-ai-systems/
28. A Primer on Large Language Models and their Limitations - arXiv, 4月 22, 2025にアクセス、https://arxiv.org/html/2412.04503v1
29. 【2024年最新】生成AIの問題事例4選 | 情報漏洩から著作権まで - メタバース総研, 4月 22, 2025にアクセス、https://metaversesouken.com/ai/generative_ai/trouble-cases/
30. Explainable AI (XAI): Decoding AI Decision-Making | Black Box Problem - Posos, 4月 22, 2025にアクセス、https://www.posos.co/blog-articles/explainable-ai-part-1-understanding-how-ai-makes-decisions
31. AI Cybersecurity Best Practices: Meeting a Double-Edged Challenge - Ivanti, 4月 22, 2025にアクセス、https://www.ivanti.com/blog/ai-cybersecurity-best-practices-meeting-a-double-edged-challenge
32. 7 actions that enforce responsible AI practices - Huron Consulting, 4月 22, 2025にアクセス、

https://www.huronconsultinggroup.com/insights/seven-actions-enforce-AI-practices
33. Overcoming Creative Block with AI: Proven Strategies - Salina, 4月 22, 2025にアクセス、https://salina.app/blog/overcoming-creative-block/
34. AI Tools for Writing Free: Overcoming Writer's Block - Yomu AI, 4月 22, 2025にアクセス、https://www.yomu.ai/blog/ai-tools-for-writing-free-overcoming-writers-block
35. Creative Block Breaker AI Agent | ClickUp™, 4月 22, 2025にアクセス、https://clickup.com/p/ai-agents/creative-block-breaker
36. Breaking Through Creative Block for UX Designers with AI - Mitra Innovation, 4月 22, 2025にアクセス、https://mitrai.com/ai/breaking-through-creative-block-for-ux-designers-with-ai/
37. The ChatGPT Fact-Check: exploiting the limitations of generative AI to develop evidence-based reasoning skills in college science courses - American Journal of Physiology, 4月 22, 2025にアクセス、https://journals.physiology.org/doi/10.1152/advan.00142.2024
38. Effectiveness of AI in Fact Checking: Distinguishing Fact from Fiction - LongShot AI, 4月 22, 2025にアクセス、https://www.longshot.ai/blog/ai-fact-checkers
39. 【初心者でもわかる】LLM（大規模言語モデル）とは？わかりやすく解説！ - Rabiloo（ラビロー）, 4月 22, 2025にアクセス、https://rabiloo.co.jp/blog/llm
40. 生成AIのハルシネーションはなぜ発生する？原因と即実践できる対策を解説, 4月 22, 2025にアクセス、https://officebot.jp/columns/basic-knowledge/hallucination-strategy/
41. How AI-Powered Fact-Checking Can Help Combat Misinformation | Ivy Exec, 4月 22, 2025にアクセス、https://ivyexec.com/career-advice/2025/how-ai-powered-fact-checking-can-help-combat-misinformation
42. “Don't Believe Everything You Read Online”: How AI Fact-Checking Could Challenge Political Bias in Science Information Processing - UF College of Journalism and Communications, 4月 22, 2025にアクセス、https://www.jou.ufl.edu/insights/dont-believe-everything-you-read-online-how-ai-fact-checking-could-challenge-political-bias-in-science-information-processing/
43. 【論文瞬読】大規模言語モデルの推論能力の限界：常識問題で明らかになった意外な弱点 | AI Nest - note, 4月 22, 2025にアクセス、https://note.com/ainest/n/n8c50deb45371
44. GAIA：新しいベンチマークが明らかにした大規模言語モデルの限界 | AI-SCHOLAR, 4月 22, 2025にアクセス、https://ai-scholar.tech/large-language-models/gaia
45. Human-in-the-loop - GitHub Pages, 4月 22, 2025にアクセス、https://langchain-ai.github.io/langgraph/concepts/human_in_the_loop/
46. What Is Human-in-the-Loop? A Simple Guide to this AI Term, 4月 22, 2025にアクセス、https://careerfoundry.com/en/blog/data-analytics/human-in-the-loop/
47. What is Human-in-the-Loop (HITL) in AI & ML? - Google Cloud, 4月 22, 2025にアクセス、https://cloud.google.com/discover/human-in-the-loop
48. Human-in-the-Loop in Machine Learning: What is it and How Does it Work? -

Levity.ai, 4月 22, 2025にアクセス、https://levity.ai/blog/human-in-the-loop
49. Humans in the Loop: The Design of Interactive AI Systems | Stanford HAI, 4月 22, 2025にアクセス、https://hai.stanford.edu/news/humans-loop-design-interactive-ai-systems
50. The Complete Guide to Human-in-the-Loop Automation - Klippa, 4月 22, 2025にアクセス、https://www.klippa.com/en/blog/information/human-in-the-loop/
51. Human-In-The-Loop | The Critical Role Of People In AI Tech, 4月 22, 2025にアクセス、https://userway.org/blog/human-in-the-loop/
52. Human in the Loop is Essential for AI-Driven Compliance | RadarFirst, 4月 22, 2025にアクセス、https://www.radarfirst.com/blog/why-a-human-in-the-loop-is-essential-for-ai-driven-privacy-compliance/
53. Human-in-the-loop AI: 4 best practices for workflow automation | Tines, 4月 22, 2025にアクセス、https://www.tines.com/blog/humans-in-the-loop-of-ai/
54. What is Human-in-the-Loop Cybersecurity and Why Does it Matter? - CYDEF, 4月 22, 2025にアクセス、https://cydef.io/resources/what-is-human-in-the-loop-cybersecurity-and-why-does-it-matter/
55. The Ethics of Human-in-the-Loop AI Systems in Medicine - SmartDev, 4月 22, 2025にアクセス、https://smartdev.com/human-in-the-loop-ai-systems/
56. Buildtime and Runtime Human-in-the-Loop AI (HITL) - CopilotKit, 4月 22, 2025にアクセス、https://www.copilotkit.ai/blog/buildtime-and-runtime
57. Data Privacy In AI-Driven Learning And Ethical Considerations - eLearning Industry, 4月 22, 2025にアクセス、https://elearningindustry.com/ensuring-data-privacy-and-ethical-considerations-in-ai-driven-learning
58. Secure “Human in the Loop” Interactions for AI Agents | Auth0, 4月 22, 2025にアクセス、https://auth0.com/blog/secure-human-in-the-loop-interactions-for-ai-agents/
59. Build Your First Human-in-the-Loop AI Agent with NVIDIA NIM, 4月 22, 2025にアクセス、https://developer.nvidia.com/blog/build-your-first-human-in-the-loop-ai-agent-with-nvidia-nim/
60. Model Context Protocol (MCP) - LangChain4j, 4月 22, 2025にアクセス、https://docs.langchain4j.dev/tutorials/mcp/
61. The official Python SDK for Model Context Protocol servers and clients - GitHub, 4月 22, 2025にアクセス、https://github.com/modelcontextprotocol/python-sdk
62. Supercharge VSCode GitHub Copilot using Model Context Protocol (MCP) - Easy Setup Guide - DEV Community, 4月 22, 2025にアクセス、https://dev.to/pwd9000/supercharge-vscode-github-copilot-using-model-context-protocol-mcp-easy-setup-guide-371e
63. Human-in-the-Loop — AutoGen - Microsoft Open Source, 4月 22, 2025にアクセス、https://microsoft.github.io/autogen/stable//user-guide/agentchat-user-guide/tutorial/human-in-the-loop.html

64. A Comprehensive Guide to AI Tech Stack - Sparx IT Solutions, 4月 22, 2025にアクセス、https://www.sparxitsolutions.com/blog/ai-tech-stack/
65. The Best Tech Stacks for AI-Powered Applications in 2025 - DEV Community, 4月 22, 2025にアクセス、https://dev.to/elliot_brenya/the-best-tech-stacks-for-ai-powered-applications-in-2025-efe
66. AI Tech Stack: A Complete Guide to Data, Frameworks, MLOps - Coherent Solutions, 4月 22, 2025にアクセス、https://www.coherentsolutions.com/insights/overview-of-ai-tech-stack-components-ai-frameworks-mlops-and-ides
67. AI Tech Stack: Choosing the Right Technology for Your Software - Appinventiv, 4月 22, 2025にアクセス、https://appinventiv.com/blog/choosing-the-right-ai-tech-stack/
68. Ultimate AI Agent Technology Stack Guide 2025 - Rapid Innovation, 4月 22, 2025にアクセス、https://www.rapidinnovation.io/post/ai-agent-technology-stack-recommender
69. Unpacking the Security Risks of Model Context Protocol (MCP) Servers - Upwind, 4月 22, 2025にアクセス、https://www.upwind.io/feed/unpacking-the-security-risks-of-model-context-protocol-mcp-servers
70. Model Context Protocol Flaw Allows Attackers to Compromise Victim Systems - GBHackers, 4月 22, 2025にアクセス、https://gbhackers.com/model-context-protocol-flaw/
71. The Security Risks of Model Context Protocol (MCP), 4月 22, 2025にアクセス、https://www.pillar.security/blog/the-security-risks-of-model-context-protocol-mcp
72. Model Context Protocol has prompt injection security problems - Simon Willison's Weblog, 4月 22, 2025にアクセス、https://simonwillison.net/2025/Apr/9/mcp-prompt-injection/
73. The most complete (and easy) explanation of MCP vulnerabilities I've seen so far. - Reddit, 4月 22, 2025にアクセス、https://www.reddit.com/r/AI_Agents/comments/1k15pma/the_most_complete_and_easy_explanation_of_mcp/