SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMAKURU-572103

(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

Project Report on

"Kuma: AI-Powered Personal Assistant"

submitted in partial fulfillment of the requirement for the completion of V semester of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

Suraj Kumar (1SI23AD057)

Himanshu Rai (1SI23AD016)

Aditya Raj (1SI23CS008)

under the guidance of

Dr Sheela S

Assistant Professor

Department of Computer Science and Engineering

SIT, Tumakuru-03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2025-26

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMAKURU-572103

(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the mini project work entitled "KUMA: AI-POWERED PERSONAL ASSISTANT" is a bonafide work carried out by Suraj Kumar (1SI23AD057), Himanshu Rai (1SI23AD016) and Aditya Raj (1SI23CS008) in partial fulfillment for the completion of V Semester of Bachelor of Engineering in Artificial Intelligence and Data Science from Siddaganga Institute of Technology, an autonomous institute under Visvesvaraya Technological University, Belagavi during the academic year 2025-26. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The Mini project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering degree.

Dr Sheela S
Assistant Professor
Dept. of CSE
SIT,Tumakuru-03

Head of the Department
Dept. of CSE
SIT,Tumakuru-03

External viva:

Names of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

We offer our humble pranams at the lotus feet of **His** Holiness, **Dr. Sree Sree Sivakumara Swamigalu**, Founder President and **His** Holiness, **Sree Sree Siddalinga Swamigalu**, President, Sree Siddaganga Education Society, Sree Siddaganga Math for bestowing upon their blessings.

We deem it as a privilege to thank **Dr. Shivakumaraiah**, CEO, SIT, Tumakuru, and **Dr. S V Dinesh**, Principal, SIT, Tumakuru for fostering an excellent academic environment in this institution, which made this endeavor fruitful.

We would like to express our sincere gratitude to **Dr Sunitha. N R**, Professor and Head, Department of Computer Science and Engineering, SIT, Tumakuru for her encouragement and valuable suggestions.

We thank our guide **Dr Sheela S**, Assistant Professor, Department of Computer Science and Engineering, SIT, Tumakuru for the valuable guidance, advice and encouragement.

Suraj Kumar (1SI23AD057)
Himanshu Rai (1SI23AD016)
Aditya Raj (1SI23CS008)

Course Outcomes

CO1: To identify a problem through literature survey and knowledge of contemporary engineering technology.

CO2: To consolidate the literature search to identify issues/gaps and formulate the engineering problem

CO3: To prepare project schedule for the identified design methodology and engage in budget analysis, and share responsibility for every member in the team

CO4: To provide sustainable engineering solution considering health, safety, legal, cultural issues and also demonstrate concern for environment

CO5: To identify and apply the mathematical concepts, science concepts, engineering and management concepts necessary to implement the identified engineering problem

CO6: To select the engineering tools/components required to implement the proposed solution for the identified engineering problem

CO7: To analyze, design, and implement optimal design solution, interpret results of experiments and draw valid conclusion

CO8: To demonstrate effective written communication through the project report, the one-page poster presentation, and preparation of the video about the project and the four page IEEE/Springer/ paper format of the work

CO9: To engage in effective oral communication through power point presentation and demonstration of the project work

CO10:To demonstrate compliance to the prescribed standards/ safety norms and abide by the norms of professional ethics

CO11: To perform in the team, contribute to the team and mentor/lead the team

CO-PO Mapping

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSo3
CO-1											3		3	
CO-2		3		3									3	
CO-3										3	3		3	
CO-4						3	3						3	
CO-5	3	3											3	
CO-6					3								3	
CO-7		3	3	3									3	
CO-8									3				3	
CO-9									3				3	
CO-10							3						3	
CO-11								3					3	

Attainment level: - 1: Slight (low) 2: Moderate (medium) 3: Substantial (high) POs: PO1: Engineering Knowledge, PO2: Problem analysis, PO3: Design/Development of solutions, PO4: Conduct investigations of complex problems, PO5: Engineering tool usage, PO6: Engineer and the world, PO7: Ethics, PO8: Individual and collaborative team work, PO9: Communication, PO10: Project management and finance, PO11: Lifelong learning PSO1:Computer based systems development,PSO2:Software development,PSO3:Computer Communications and Internet applications

Abstract

[Add abstract here covering three paragraphs:

Paragraph 1 - Motivation: Discuss the growing need for intelligent personal assistants that can handle diverse tasks across productivity tools, voice interaction, and document analysis. Highlight limitations of existing solutions and the opportunity for a customizable, self-hosted alternative.

Paragraph 2 - Objectives: State the main goals including development of a multi-agent AI system, voice-enabled interaction with Indic language support, vision capabilities for image and document analysis, scalable architecture using Redis message queues, and containerized deployment using Docker.

Paragraph 3 - Implementation: Mention key technologies: TypeScript and Bun runtime for backend, React and Vite for frontend, Google Gemini and OpenAI for AI processing, Sarvam AI for voice, Redis Streams for message queuing, PostgreSQL with Prisma ORM for data persistence, and Docker for containerized deployment. Highlight the multi-agent architecture with specialized agents for different task domains.]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

[Discuss the following points:

- Growing complexity of digital workflows requiring intelligent automation
- Fragmentation of productivity tools (email, calendar, documents) needing unified access
- Limitations of existing assistants: closed ecosystems, limited customization, privacy concerns
- Emergence of powerful LLMs enabling sophisticated conversational agents
- Need for multimodal interaction combining text, voice, and visual inputs
- Importance of self-hosted solutions for data privacy and customization
- Opportunity to leverage open-source AI frameworks for building personalized assistants
- Growing demand for Indic language support in voice-based interfaces

]

1.2 Objective of the project

[State the primary objectives of the Kuma project:

- Development of an intelligent personal assistant using modern AI technologies
- Implementation of multi-agent architecture with specialized agents for different tasks
- Voice-enabled interaction with speech-to-text and text-to-speech capabilities
- Image understanding and vision-based document analysis

- Integration of productivity services (Gmail, Calendar, Docs, Drive, GitHub)
- Scalable message queue architecture for reliable AI processing
- Containerized deployment using Docker for production readiness
- Creating a responsive and intuitive user interface

]

1.3 Organisation of the report

[Describe the structure of the report:

- Chapter 1: Introduction covering motivation, objectives, and report organization
- Chapter 2: Literature survey on conversational AI, agent frameworks, voice processing, and containerization
- Chapter 3: System design including architecture, data flow, Redis queue system, and Docker deployment
- Chapter 4: Implementation covering backend services, AI agents, voice pipeline, vision processing, and frontend
- Chapter 5: Results with screenshots, performance benchmarks, and comparative analysis
- Chapter 6: Conclusion summarizing achievements and outlining future enhancements
- Bibliography and Appendices with SDG mapping and component data sheets

]

Chapter 2

Literature Survey

2.1 Conversational AI and Virtual Assistants

[Review existing AI assistants - Google Assistant, Siri, Alexa, ChatGPT. Discuss their capabilities, limitations, and architectural approaches. Analyze trends in conversational AI research.]

2.2 Large Language Models and Agent Frameworks

[Explore LLM architectures and their evolution. Discuss agent frameworks including LangChain, AutoGPT, and ReAct pattern. Review multi-agent coordination strategies.]

2.3 Google Gemini and Multimodal AI

[Analyze Google Gemini's multimodal capabilities for text, image, and audio processing. Compare with GPT-4V and other vision-language models.]

2.4 Speech Processing Technologies

[Review speech-to-text and text-to-speech technologies. Discuss real-time voice communication protocols and WebRTC/LiveKit frameworks. Analyze Indic language support in voice systems.]

2.5 Image Understanding and Vision AI

[Explore computer vision for document analysis. Review OCR technologies and vision-language models for image captioning and visual question answering.]

2.6 Message Queue Architectures

[Discuss asynchronous processing patterns using Redis Streams, RabbitMQ, and Kafka. Analyze producer-consumer models and dead letter queue strategies.]

2.7 Containerization and Deployment

[Review Docker containerization benefits and multi-stage build patterns. Discuss container orchestration and microservices deployment strategies.]

2.8 Web Application Technologies

[Analyze modern web stack choices - React, TypeScript, Vite for frontend. Review Bun runtime, Express, and Prisma ORM for backend development. Mention JWT for user sessions and OAuth 2.0 for third-party service integration.]

[Include citations from IEEE, Springer, ACM, and other peer-reviewed journals. Use proper citation format: [1], [2].]

Chapter 3

System Design & Methodology

3.1 Functional & Non-Functional Requirements

3.1.1 Functional Requirements

[List functional requirements:

1. User login and session management
2. Real-time chat interface with streaming AI responses
3. Multi-agent system with intelligent routing to specialized agents
4. Voice input processing with speech-to-text transcription
5. Voice output generation with text-to-speech synthesis
6. Image upload and vision-based analysis
7. Document upload with PDF text extraction and summarization
8. Integration with Google services (Gmail, Calendar, Docs, Drive)
9. GitHub repository interaction capabilities
10. Long-term memory storage and contextual retrieval
11. Web search and information gathering tools
12. Asynchronous message processing via Redis queue
13. Docker-based containerized deployment

]

3.1.2 Non-Functional Requirements

[List non-functional requirements:

1. Performance: AI response generation within 3 seconds, voice latency under 500ms
2. Security: Basic user authentication and secure API access
3. Scalability: Horizontal scaling via Redis queue and containerized deployment
4. Reliability: Health checks, automatic container restarts, dead letter queue for failed jobs
5. Usability: Responsive interface supporting both text and voice input
6. Maintainability: Modular architecture with separate services for API, worker, and frontend
7. Portability: Docker containers enabling consistent deployment across environments

]

3.2 List of Hardware & Software Requirements

3.2.1 Hardware Requirements

[Specify hardware requirements:

- Processor: Intel Core i5 or equivalent (minimum)
- RAM: 8 GB (minimum), 16 GB (recommended)
- Storage: 20 GB free space
- Network: Stable internet connection

]

3.2.2 Software Requirements

[Specify software requirements:

- Operating System: Windows 10/11, macOS, or Linux

- Runtime: Bun ¿= 1.0.0
- Database: PostgreSQL
- Node.js: Version 18+ (optional)
- Browser: Chrome, Firefox, Safari (latest versions)
- Development Tools: VS Code or similar IDE
- Version Control: Git

Key Technologies:

- Backend: TypeScript, Bun runtime, Express framework, Prisma ORM
- Frontend: React 18, Vite, TypeScript, Zustand state management, shadcn/ui components
- AI/ML: Vercel AI SDK, Google Gemini API, OpenAI API, Supermemory for long-term memory
- Voice: Sarvam AI for Indic speech-to-text and text-to-speech, LiveKit for real-time communication
- Vision: Google Gemini Vision for image analysis and document understanding
- Queue: Redis Streams for asynchronous message processing
- Containerization: Docker, Docker Compose, NGINX for production serving
- APIs: Google OAuth 2.0, Gmail API, Calendar API, Drive API, Docs API, GitHub API

]

3.3 System Architecture

[Describe the overall system architecture. Include:

- Three-tier architecture with React frontend, Express API, and PostgreSQL database
- Client-Server communication via REST API and Server-Sent Events for streaming

- Redis-based message queue for asynchronous AI processing
- Multi-agent system with router agent delegating to specialized agents
- Voice pipeline connecting Sarvam STT/TTS with AI agents
- Vision module for image analysis and document processing
- OAuth integration layer for Google and GitHub services
- Docker containerization with separate containers for API, worker, and frontend

Refer to Figure 3.1 for the system architecture diagram.]

Figure 3.1: System Architecture of Kuma AI Assistant

3.4 Redis Queue Architecture

[Describe the asynchronous message processing system:

- Producer-consumer pattern using Redis Streams
- Message job lifecycle: pending, processing, completed, failed
- Consumer groups for reliable message delivery
- Dead letter queue for failed message handling
- Status tracking and real-time updates via pub/sub

Refer to Figure 3.2 for the Redis queue flow diagram.]

Figure 3.2: Redis Message Queue Architecture

3.5 Voice Processing Architecture

[Describe the voice interaction pipeline:

- Audio capture and streaming using LiveKit
- Speech-to-text conversion using Sarvam AI
- AI agent processing of transcribed text

- Text-to-speech synthesis for voice responses
- Real-time bidirectional communication flow

Refer to Figure 3.3 for the voice processing flow diagram.]

Figure 3.3: Voice Processing Pipeline

3.6 Docker Deployment Architecture

[Describe the containerized deployment setup:

- Multi-stage Dockerfile builds for optimized images
- Docker Compose orchestration of all services
- Service containers: frontend (NGINX), backend (Bun), worker, PostgreSQL, Redis
- Volume mounts for persistent data storage
- Health checks and automatic restart policies
- Network isolation using Docker bridge networks

Refer to Figure 3.4 for the Docker deployment diagram.]

Figure 3.4: Docker Deployment Architecture

3.7 Data Flow Diagrams

3.7.1 Level 0 DFD - Context Diagram

[Describe the context-level data flow showing the system as a single process with external entities:

- User (text, voice, image inputs)
- Google Services (Gmail, Calendar, Docs, Drive)
- AI APIs (Gemini, OpenAI, Supermemory)
- Voice Services (Sarvam STT/TTS, LiveKit)

]

Figure 3.5: Context Diagram (Level 0 DFD)

3.7.2 Level 1 DFD

[Describe the Level 1 DFD showing major processes:

- Authentication and Session Management
- Chat and Message Processing
- AI Agent Routing and Execution
- Voice Input/Output Processing
- Image and Document Analysis
- External Service Integration
- Redis Queue Management

]

Figure 3.6: Level 1 Data Flow Diagram

3.7.3 Level 2 DFD - Agent Processing

[Describe detailed data flow within the agent processing module:

- Router agent receiving user query
- Agent selection based on query classification
- Tool invocation and external API calls
- Response generation and formatting
- Memory storage and context retrieval

]

Figure 3.7: Level 2 DFD - Agent Processing Module

3.7.4 Level 2 DFD - Voice Processing

[Describe detailed data flow for voice interactions:

- Audio stream capture and buffering
- Speech recognition and transcription
- Text processing by AI agent
- Speech synthesis and audio streaming

]

Figure 3.8: Level 2 DFD - Voice Processing Module

3.7.5 Level 2 DFD - Image Processing

[Describe detailed data flow for vision capabilities:

- Image upload and validation
- Vision model analysis (Gemini Vision)
- Text extraction (OCR) from documents
- Scene description and object detection
- Response generation with visual context

]

Figure 3.9: Level 2 DFD - Image Processing Module

3.8 Algorithms

3.8.1 Chat Processing Algorithm

[Describe the algorithm for processing user queries:

1. Receive user input (text, images, or documents)
2. Validate JWT token and authenticate user session

3. Create or retrieve existing chat thread
4. Load conversation context and relevant memories
5. Route to appropriate specialized agent
6. Execute agent with tool access and external APIs
7. Stream response tokens via Server-Sent Events
8. Persist message and update conversation summary
9. Return complete response to user

]

3.8.2 Agent Selection Algorithm

[Describe the router agent's decision process:

1. Analyze user query semantics and intent
2. Check for domain-specific keywords (email, calendar, code, etc.)
3. Evaluate available agent capabilities
4. Select most appropriate specialized agent
5. Fallback to general assistant for ambiguous queries

]

3.8.3 Redis Queue Processing Algorithm

[Describe the asynchronous message processing flow:

1. Producer publishes message job to Redis Stream
2. Consumer group claims pending messages
3. Worker processes message through AI agent pipeline
4. Status updates published via Redis pub/sub
5. Completed response stored in database

6. Failed messages moved to dead letter queue after retries
7. Client receives updates through SSE subscription

]

3.8.4 Voice Processing Algorithm

[Describe the voice interaction pipeline:

1. Capture audio stream from client microphone
2. Buffer audio chunks for processing
3. Send audio to Sarvam STT for transcription
4. Process transcribed text through AI agent
5. Convert AI response to speech using Sarvam TTS
6. Stream synthesized audio back to client
7. Handle interruptions and turn-taking

]

3.8.5 Image Analysis Algorithm

[Describe the vision processing workflow:

1. Receive image upload with optional prompt
2. Validate file type and size constraints
3. Encode image to base64 for API transmission
4. Send to Gemini Vision model with context
5. Parse structured response (description, extracted text, objects)
6. Store analysis results with chat message
7. Return formatted response to user

]

3.8.6 Memory Management Algorithm

[Describe context and memory handling:

1. Retrieve recent message history from database
2. Query Supermemory for relevant long-term memories
3. Combine into structured context for AI agent
4. After response, extract key facts for memory storage
5. Periodically summarize old conversations
6. Prune and consolidate duplicate memories

]

3.8.7 Google Services Connection Flow

[Describe the OAuth 2.0 flow for connecting Google apps:

1. User initiates connection for a Google service (Gmail, Calendar, etc.)
2. Redirect to Google consent screen with required scopes
3. User grants permissions on Google's authorization page
4. Google redirects back with authorization code
5. Exchange code for access and refresh tokens
6. Store encrypted tokens for future API calls
7. Automatically refresh tokens when expired

]

Chapter 4

Implementation Details

4.1 Backend Implementation

4.1.1 Project Setup

[Describe the backend setup using Bun, TypeScript, and Express. Include package installation, configuration, and project structure.]

4.1.2 Database Design

[Describe the Prisma schema, database tables, relationships, and migrations. Include the schema for:

- User table
- Chat table
- Message table
- Agent table
- Document table
- Memory table

]

4.1.3 API Endpoints

[Document all API endpoints:

- Authentication routes (/auth)
- Chat routes (/chat)
- Agent routes (/agents)
- Document routes (/documents)

- Upload routes (/upload)
- App integration routes (/apps)

]

4.1.4 AI Integration

[Describe the implementation of:

- Vercel AI SDK configuration for streaming responses
- Google Gemini and OpenAI model integration
- Custom tool definitions using Zod schemas
- System prompt engineering for each agent type
- Hybrid memory using chat history and Supermemory
- Multi-agent router pattern implementation

]

4.1.5 Voice Processing Implementation

[Describe the implementation of:

- Sarvam AI client for Indic language STT/TTS
- Audio format handling and conversion
- LiveKit integration for real-time voice rooms
- Token generation for secure voice sessions
- Voice-to-agent pipeline coordination
- Error handling for audio processing failures

]

4.1.6 Vision and Image Processing

[Describe the implementation of:

- Multer configuration for image uploads
- Base64 encoding for API transmission
- Gemini Vision API integration
- OCR text extraction from documents
- Scene description and analysis
- Image attachment storage and retrieval

]

4.1.7 Redis Queue Implementation

[Describe the implementation of:

- Redis client connection management
- Stream producer for message publishing
- Consumer group and worker setup
- Job status tracking via pub/sub
- Dead letter queue for failed messages
- Retry logic with exponential backoff
- Health monitoring and metrics collection

]

4.1.8 Google Services Integration

[Describe the implementation of:

- OAuth 2.0 flow for connecting Google accounts
- Gmail API for reading, searching, and composing emails

- Google Calendar for event creation and schedule queries
- Google Docs for document creation and editing
- Google Drive for file listing and management
- Google Sheets for spreadsheet operations
- Token refresh mechanism for persistent access

]

4.1.9 Other External Services

[Describe the implementation of:

- GitHub API for repository operations and code search
- Exa for intelligent web search capabilities
- Supermemory for long-term memory storage and retrieval

]

4.1.10 Docker Containerization

[Describe the implementation of:

- Multi-stage Dockerfile for backend API
- Separate worker Dockerfile for queue processing
- Frontend Dockerfile with NGINX serving
- Docker Compose for service orchestration
- Environment variable management
- Volume configuration for persistent data
- Health check definitions
- Development vs production configurations

]

4.2 Frontend Implementation

4.2.1 Project Setup

[Describe the frontend setup using React, Vite, and TypeScript. Include component structure and routing.]

4.2.2 State Management

[Describe Zustand store implementation for managing application state]

4.2.3 UI Components

[Describe the implementation of key UI components:

- Chat interface
- Message bubbles
- File upload component
- Authentication forms
- Navigation menu
- Settings panel

]

4.2.4 API Integration

[Describe how the frontend communicates with the backend API, including error handling and loading states]

4.3 Security Implementation

[Describe security measures:

- JWT-based authentication
- Data encryption
- CORS configuration
- Input validation and sanitization

- Secure API key management

]

4.4 Code Snippets

[Include relevant code snippets for key implementations]

Chapter 5

Results

5.1 Screenshots

[Include screenshots of:

- Login and Registration pages
- Main chat interface with message history
- AI agent responses with tool usage
- Voice interaction interface
- Image upload and analysis results
- Document processing and OCR output
- Google service integrations (Gmail, Calendar)
- App connection settings
- Memory and context display
- Mobile responsive views
- Docker container status

Example references:]

Figure 5.1: Main Chat Interface

Figure 5.2: Voice Interaction Interface

Figure 5.3: Image Analysis Results

Figure 5.4: Gmail Integration

Figure 5.5: Docker Container Dashboard

5.2 Analysis

5.2.1 Performance Metrics

[Present performance analysis with tables and graphs:

- AI response generation time
- Voice transcription and synthesis latency
- Image processing duration
- Redis queue throughput
- Database query performance
- Docker container resource usage
- Concurrent user handling capacity

]

Table 5.1: Text Chat Performance Metrics

Metric	**Minimum**	**Average**	**Maximum**
AI Response Time (ms)			
First Token Latency (ms)			
Database Query (ms)			
Memory Usage (MB)			

5.2.2 Comparison with Existing Systems

[Compare Kuma with existing AI assistants across key capabilities]

5.2.3 Testing Results

[Present results from:

- Unit testing

Table 5.2: Voice Processing Metrics

Metric	**Minimum**	**Average**	**Maximum**
STT Latency (ms)			
TTS Latency (ms)			
End-to-End Voice (ms)			
Audio Quality (MOS)			

Table 5.3: Vision Processing Metrics

Metric	**Minimum**	**Average**	**Maximum**
Image Analysis (ms)			
OCR Extraction (ms)			
Scene Description (ms)			

- Integration testing
- User acceptance testing
- Performance testing

]

5.2.4 User Feedback

[If applicable, include user feedback and satisfaction metrics]

Table 5.4: Docker Resource Usage

Container	**CPU (%)**	**Memory (MB)**	**Image Size (MB)**
Backend API			
Worker			
Frontend (NGINX)			
PostgreSQL			
Redis			

Table 5.5: Comparison with Existing AI Assistants

Feature	**Kuma**	**ChatGPT**	**Google Assistant**	**Siri**	**Alexa**
Custom AI Agents	Yes	Limited	No	No	No
Multi-Agent Routing	Yes	No	No	No	No
Gmail Integration	Yes	No	Yes	No	No
Calendar Integration	Yes	No	Yes	Yes	Yes
Document Analysis	Yes	Yes	Limited	Limited	No
Voice Interaction	Yes	Yes	Yes	Yes	Yes
Image Understanding	Yes	Yes	Yes	Yes	Limited
Indic Language Voice	Yes	Limited	Yes	Limited	Limited
Self-Hosted	Yes	No	No	No	No
Open Source	Yes	No	No	No	No
Docker Deployment	Yes	N/A	N/A	N/A	N/A
Long-term Memory	Yes	Yes	Limited	Limited	Limited

Chapter 6

Conclusion & Future Enhancement

6.1 Conclusion

[Summarize the project achievements:

- Successfully developed a comprehensive AI-powered personal assistant platform
- Implemented multi-agent architecture with router pattern for intelligent task delegation
- Integrated voice interaction with Indic language support using Sarvam AI
- Built vision capabilities for image analysis and document understanding
- Designed scalable architecture with Redis message queues for reliable processing
- Containerized entire stack using Docker for consistent deployment
- Integrated multiple productivity services (Gmail, Calendar, Docs, Drive, GitHub)
- Created responsive user interface with real-time streaming responses
- Achieved production-ready deployment with health monitoring and error recovery

Highlight the learning outcomes: practical experience with modern AI frameworks, full-stack development, containerization, and building production-grade applications.]

6.2 Future Enhancement

[Discuss potential future improvements:

- Additional service integrations (Slack, Microsoft Office 365, Notion)
- Mobile application using React Native for iOS and Android
- Fine-tuned models trained on user interaction patterns
- Multi-user collaborative workspaces with shared agents

- Plugin architecture for community-developed agents and tools
- WebSocket-based real-time collaboration features
- Kubernetes deployment for enterprise-scale orchestration
- On-device voice processing for reduced latency
- Offline mode with local LLM support (Ollama integration)
- Advanced analytics dashboard for usage insights
- Biometric and multi-factor authentication options
- IoT device integration for smart home control
- Video call integration with screen sharing analysis
- Browser extension for contextual assistance

]

Bibliography

[1] LangChain Documentation, *"LangChain: Building applications with LLMs through composability"*, `https://langchain.com/docs`, 2024.

[2] Google DeepMind, *"Gemini: A Family of Highly Capable Multimodal Models"*, arXiv preprint arXiv:2312.11805, December 2023.

[3] Meta Open Source, *"React: A JavaScript library for building user interfaces"*, `https://react.dev`, 2024.

[4] Microsoft Corporation, *"TypeScript: JavaScript with syntax for types"*, `https://www.typescriptlang.org`, 2024.

[5] Prisma Data, Inc., *"Prisma: Next-generation Node.js and TypeScript ORM"*, `https://www.prisma.io`, 2024.

[6] D. Hardt, *"The OAuth 2.0 Authorization Framework"*, RFC 6749, IETF, October 2012.

[7] Redis Ltd., *"Redis Streams: Introduction to Redis Streams"*, `https://redis.io/docs/data-types/streams/`, 2024.

[8] Docker Inc., *"Docker Documentation: Build, Share, and Run Container Applications"*, `https://docs.docker.com`, 2024.

[9] Vercel Inc., *"AI SDK: The TypeScript toolkit for building AI applications"*, `https://sdk.vercel.ai/docs`, 2024.

[10] LiveKit Inc., *"LiveKit: Open source WebRTC infrastructure"*, `https://livekit.io/docs`, 2024.

[11] Sarvam AI, *"Sarvam APIs: Speech-to-Text and Text-to-Speech for Indic Languages"*, `https://www.sarvam.ai`, 2024.

[12] Supermemory, *"Supermemory: Long-term memory for AI applications"*, `https://supermemory.ai`, 2024.

[13] Oven Sh., *"Bun: A fast all-in-one JavaScript runtime"*, `https://bun.sh`, 2024.

[14] S. Yao et al., *"ReAct: Synergizing Reasoning and Acting in Language Models"*, ICLR 2023, arXiv:2210.03629.

[15] A. Vaswani et al., *"Attention Is All You Need"*, NeurIPS 2017.

Appendices

Appendix A

Sustainable Development Goals addressed

#	**SDG**	**Level**
1	No Poverty	
2	Zero Hunger	
3	Good Health and Well-being	
4	Quality education	
5	Gender Quality	
6	Clean water and Sanitation	
7	Affordable and Clean Energy	
8	Decent work and Economic Growth	
9	Industry, Innovation and Infrastructure	
10	Reduced Inequalities	
11	Sustainable cities and Communities	
12	Responsible Consumption and production	
13	Climate action	
14	Life below water	
15	Life on Land	
16	Peace, Justice and Strong Institutions	
17	Partnership's for the Goals	

Levels: Poor:1, Good :2, Excellent:3

Appendix B

Self-Assesment of the Project

#	PO and PSO	Contribution from the Project	Level
1	Engineering Knowledge:		
2	Problem Analysis:		
3	Design/development of solutions		
4	Conduct investigations of complex problems:		
5	Modern tool usage:		
6	The Engineer and the world:		
7	Ethics:		
8	Individual and Team Work:		
9	Communication:		
10	Project Management and Finance:		
11	Life-long Learning:		
1	PSO1		
2	PSO2		
3	PSO3		

PSO1: Computer based systems development: Ability to apply the basic knowledge of database systems, computing, operating system, digital circuits, microcontroller, computer organization and architecture in the design of computer based systems.

PSO2: Software development: Ability to specify, design and develop projects, application softwares and system softwares by using the knowledge of data structures, analysis and design of algorithm, programming languages, software engineering practices and open source tools.

PSO3: Computer communications

and Internet applications: Ability to design and develop network protocols and internet applications by incorporating the knowledge of computer networks, communication protocol engineering, cryptography and network security, distributed and cloud computing, data mining, big data analytics, ad hoc networks, storage area networks and wireless sensor networks.

Levels: Poor:1, Good :2, Excellent:3

Appendix C

Data Sheet of component 1

Note: Only include relevant details of the components that are referred w.r.t. project.

Appendix D

Data Sheet of component 2