Auto Return

An Agentic Unified Intelligent Multi-Platform Communication System

Project Team

Hasnain Saleem 22P-9123
Alishba Tariq 22P-9112
Kashan Saeed 22P-9128

Session 2022–2026

Supervised by

Dr. Nouman Azam (Associate Professor)

Department of Computer Science

National University of Computer and Emerging Sciences
Peshawar, Pakistan

December, 2025

Student's Declaration

We declare that this project titled "*Auto Return*
An Agentic Unified Intelligent Multi-Platform Communication System", submitted as requirement for the award of degree of Bachelors in Bachelor of Science in Computer Science, does not contain any material previously submitted for a degree in any university; and that to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the management of Department of Computer Science, National University of Computer and Emerging Sciences, has a zero tolerance policy towards plagiarism. Therefore, We, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized and any material used in the thesis from other sources is properly referenced.

We further understand that if we are found guilty of any form of plagiarism in the thesis work even after graduation, the University reserves the right to revoke our BS degree.

Hasnain Saleem Signature: _____

Alishba Tariq Signature: _____

Kashan Saeed Signature: _____

_____

Verified by Plagiarism Cell Officer

Dated:

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Auto Return*
An Agentic Unified Intelligent Multi-Platform Communication System, submitted by Hasnain Saleem (22P-9123), Alishba Tariq (22P-9112), and Kashan Saeed (22P-9128), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Bachelor of Science in Computer Science.

Supervisor

Dr. Nouman Azam (Associate Professor) Signature: _____

Sir Riaz Nawab

FYP Coordinator
National University of Computer and Emerging Sciences, Peshawar

---

Dr. Qasm Jan, Head of Computer Science Department

HoD of Department of Computer Science
National University of Computer and Emerging Sciences

Acknowledgements

We would like to express our deepest gratitude to our supervisor, **Dr. Nouman Azam**, for his continuous guidance, motivation, and valuable knowledge throughout the development of our Final Year Project. We also thank our department faculty and peers for their constructive feedback and encouragement. Lastly, we appreciate our families for their constant support and patience during the completion of this project.

Hasnain Saleem

Alishba Tariq

Kashan Saeed

Abstract

This project presents **Auto Return**, a voice-first, local-LLM-powered automation platform built on the FORGE Framework, designed to unify communication and task management tools (Gmail, Slack, Jira) through an intelligent orchestrator. The system addresses the fragmentation of productivity tools by providing a unified intelligent workspace with voice-enabled automation, smart agents that summarize, reply, and prioritize automatically, and a centralized notification hub.

The core technical components include: 1) A Python-native, async-first FORGE Framework with event-driven architecture; 2) Local LLM integration using Ollama with Llama 3.1 8B; 3) Agent-based architecture with Gmail, Slack, and Jira integrations; 4) Custom priority classification algorithm based on urgency, deadlines, and sender importance; 5) Multiple intelligent pipelines for summarization, draft generation, task extraction, and file attachment.

Implementation status shows significant progress with 10+ features implemented including Gmail/Slack integration, draft generation, email summarization, task extraction, and unified inbox (80% complete). The system demonstrates enhanced productivity, reduced cognitive load, and effective cross-platform communication through automation while maintaining local-first privacy.

Contents

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Project Overview

Auto Return is a voice-first, local-LLM-powered automation platform built on the FORGE Framework, designed to unify communication and task management tools (Gmail, Slack, Jira) through an intelligent orchestrator. The system operates as a native desktop application built with PyQt6 (not a web wrapper) and supports cross-platform deployment (Linux, Windows, macOS).

1.2 Motivation

- Managing multiple apps (Gmail, Slack, terminal) separately causes frequent context switching.
- Repetitive communication and task handling done manually reduces productivity.
- Notifications are scattered and unmanageable across different platforms.
- Need for intelligent automation that can summarize, prioritize, and respond automatically.
- Growing demand for local-first AI solutions that preserve privacy and work offline.

1.3 Problem Statement

The absence of an integrated, voice-driven, AI-powered automation platform causes inefficiency, fragmented user experiences, and reduced productivity due to constant context switching between disparate communication and task management tools.

1.4 Proposed Solution & Objectives

Auto Return introduces a unified intelligent workspace integrating all tools with voice-enabled automation. The objectives are:

- Provide unified inbox aggregating messages from Gmail and Slack.
- Implement voice interface with wake word activation ("Hey Auto").
- Automate task extraction and create Jira tickets automatically.
- Generate intelligent summaries and drafts using local LLM.
- Implement custom priority classification for emails/messages.
- Ensure cross-platform support (Linux, Windows, macOS).
- Maintain local-first privacy with SQLite-based storage.

Chapter 2

System Analysis and Design

2.1 System Architecture

The system follows the **FORGE Framework** - a Python-native, async-first framework with event-driven architecture using pub/sub pattern. Key architectural components include:

- **Local LLM Integration:** Ollama with Llama 3.1 8B for privacy-preserving AI processing.
- **Agent-based Architecture:** Modular agents for Gmail, Slack, and Jira integrations.
- **Extensible Plugin System:** For adding new integrations and functionality.
- **SQLite-based Memory:** With vector storage for intelligent retrieval.
- **Native Desktop App:** Built with PyQt6 (not a web wrapper).

2.2 Core Components

2.2.1 FORGE Framework

- Type: Python-native, async-first framework

- Architecture: Event-driven with pub/sub pattern
- Key Features: Local LLM integration, agent-based architecture, extensible plugin system, SQLite-based memory and vector storage

2.2.2 Main Features

- Voice Interface: Wake word activation ("Hey Auto") with natural language processing
- Unified Inbox: Aggregates messages from Gmail and Slack
- Task Automation: Auto-extracts tasks and creates Jira tickets
- Native Desktop App: Built with PyQt6
- Cross-Platform: Supports Linux, Windows, and macOS

2.3 Email/Message Priority Classification Algorithm

The system implements a custom priority classification algorithm that considers multiple factors to determine message urgency.

2.3.1 Algorithm Overview

2.3.2 Keyword Score Calculation

$$\text{Keyword\_Score}(Message) = \sum_{i=1}^{s} \text{Keyword\_Value}(K[i])$$

Where $K[1..s]$ are extracted keywords from message subject and body, categorized as:

- Direct Urgency Words (e.g., "urgent", "ASAP")
- Time Pressure Words (e.g., "deadline", "due")
- Action Call Words (e.g., "action required", "please respond")

Algorithm 1 Email/Message Priority Classification

Require: Priority thresholds θ_h (high) and θ_l (low)
Require: Weights w_1, w_2, w_3 for keyword, deadline, and sender scores
for each Message in inbox **do**
    $keyword \leftarrow$ Keyword_Score$(Message)$
    $deadline \leftarrow$ Deadline_Score$(Message)$
    $sender \leftarrow$ Sender_Score$(Message)$
    $urgency \leftarrow w_1 \times keyword + w_2 \times deadline + w_3 \times sender$
    if $urgency \geq \theta_h$ **then**
        return "High"
    else if $urgency > \theta_l$ **then**
        return "Medium"
    else
        return "Low"
    end if
end for

2.3.3 Deadline Score Calculation

$$\text{Deadline\_Score}(Message) = \text{absolute\_score} + \text{relative\_score}$$

With time-based adjustments:

- If time remaining < 24 hours: $deadline\_total \leftarrow deadline\_total + \delta/2$
- If $deadline\_total > \delta$: $deadline\_total \leftarrow \delta$ (capped maximum)

2.3.4 Sender Score Calculation

$$\text{Sender\_Score}(Message) = cu_1 \times \text{sender\_score} + cu_2 \times \text{cc\_total}$$

Where sender scores are user-defined (e.g., boss = 10, colleague = 5, unknown = 1).

2.4 Intelligent Pipelines

2.4.1 AI Summary Pipeline

1. Detect and fetch incoming email/message

2. Generate structured summary prompt for LLM using extracted message fields
3. Store generated summary in database and display in unified inbox

2.4.2 Task Extraction Pipeline

1. Fetch and extract message content from incoming email or message
2. Send extracted text to LLM to classify Action Type and Task Directive
3. Output classification to trigger next relevant pipeline

2.4.3 Plain Text Response Pipeline

1. Generate simple text reply via LLM using extracted details from email/message
2. Show reply for user review and approval
3. Send approved reply and log for records

2.4.4 Draft Generation Pipeline

1. Generate structured prompt for LLM using fetched relevant context
2. Display generated draft for editing or approval
3. Send approved draft and log for records

2.4.5 File Attachment Pipeline

1. Search local directories based on requested file name and type in message
2. Let user select file to attach among suggested matching files
3. Generate email body via LLM and show for approval or editing
4. Send email with attachment and log the action

2.5 Use Case Diagram

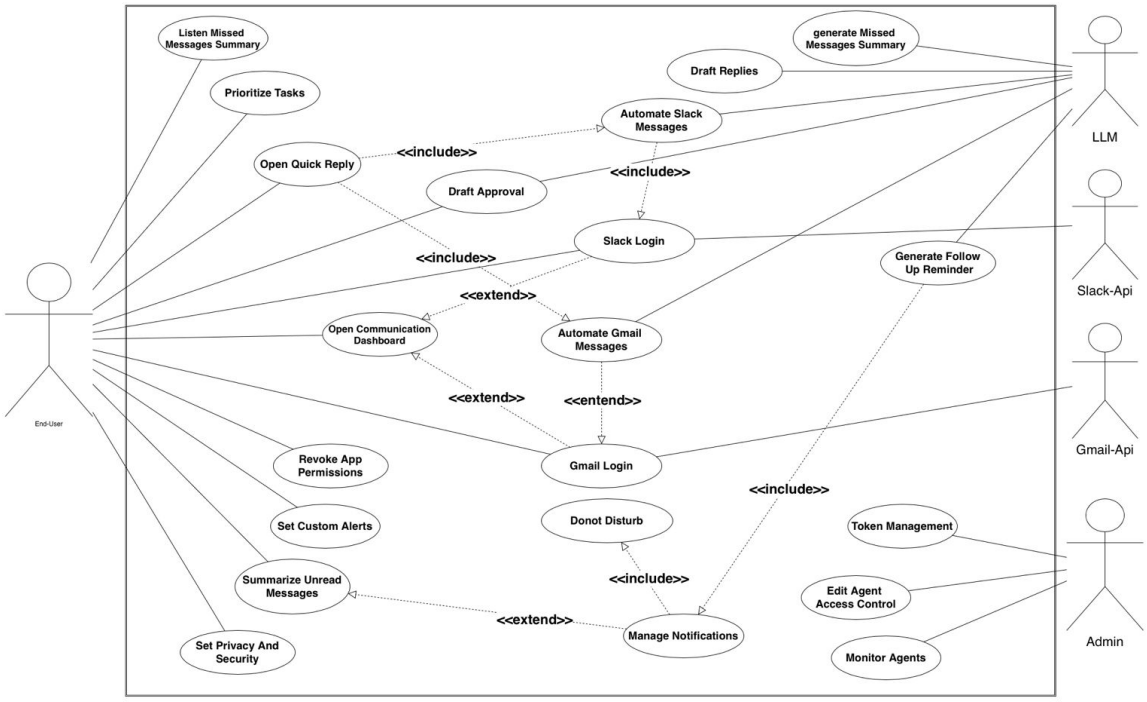

Figure 2.1: Use Case Diagram for Auto Return

2.6 Activity Diagrams

2.6.1 Overall System Activity

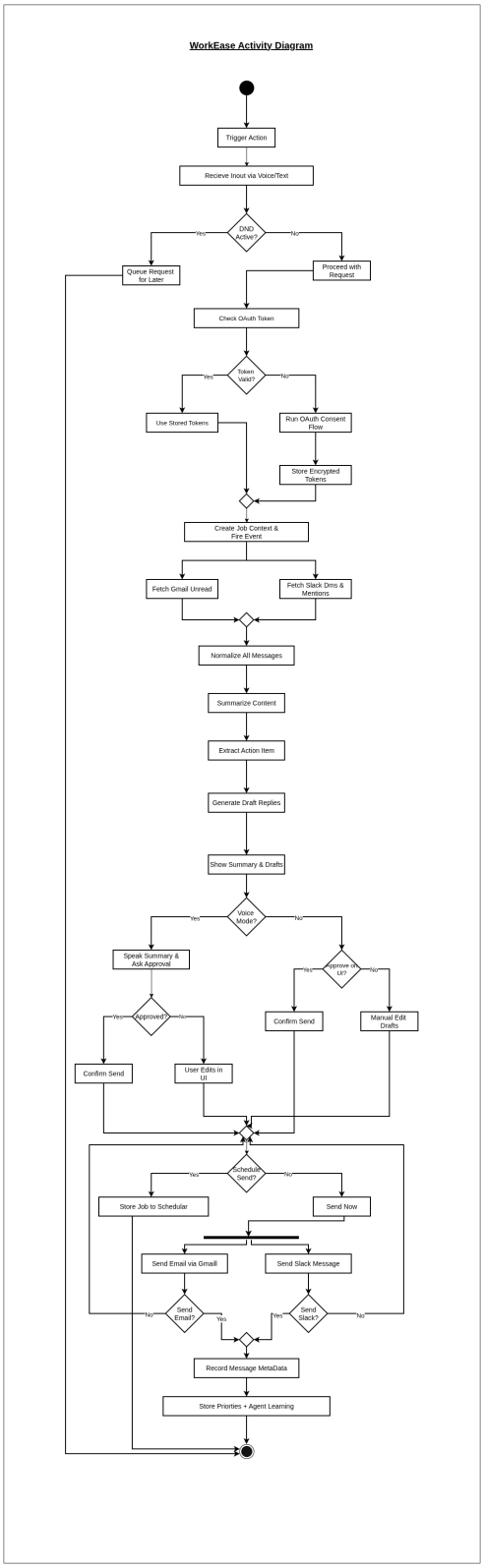

Figure 2.2: Overall System Activity Diagram

2.6.2 Activity 1: Voice-Based Interaction And Control

Figure 2.3: Activity Diagram – Voice-Based Interaction And Control

2.6.3 Activity 2: Smart Email and Message Automation

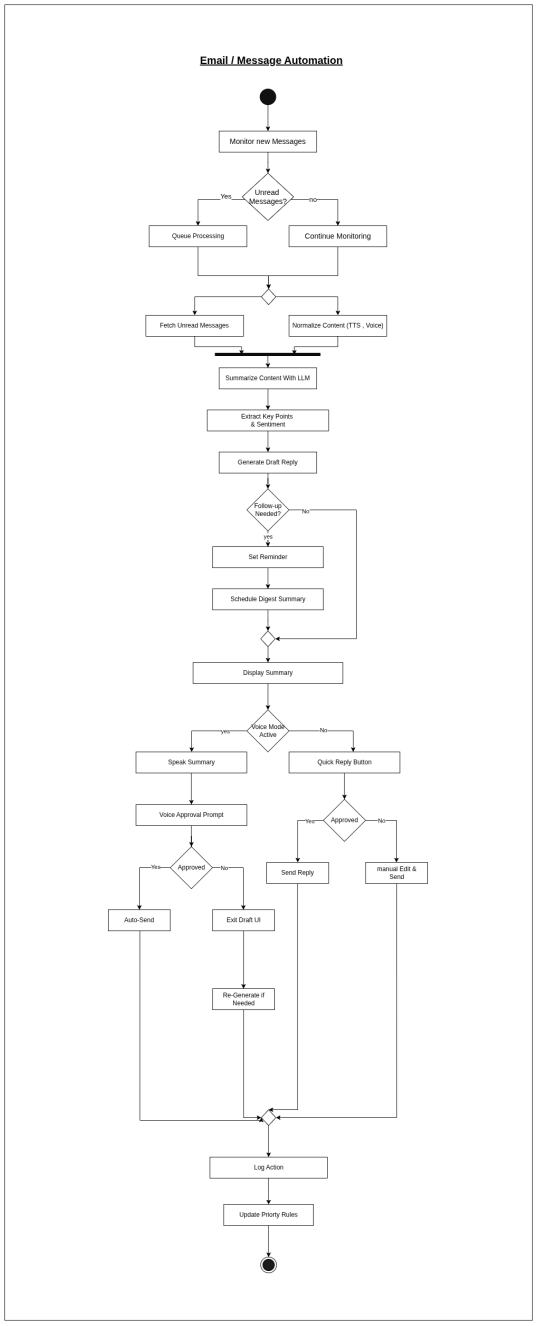

Figure 2.4: Activity Diagram – Smart Email and Message Automation

2.6.4 Activity 3: Multi-App Integration and Authentication

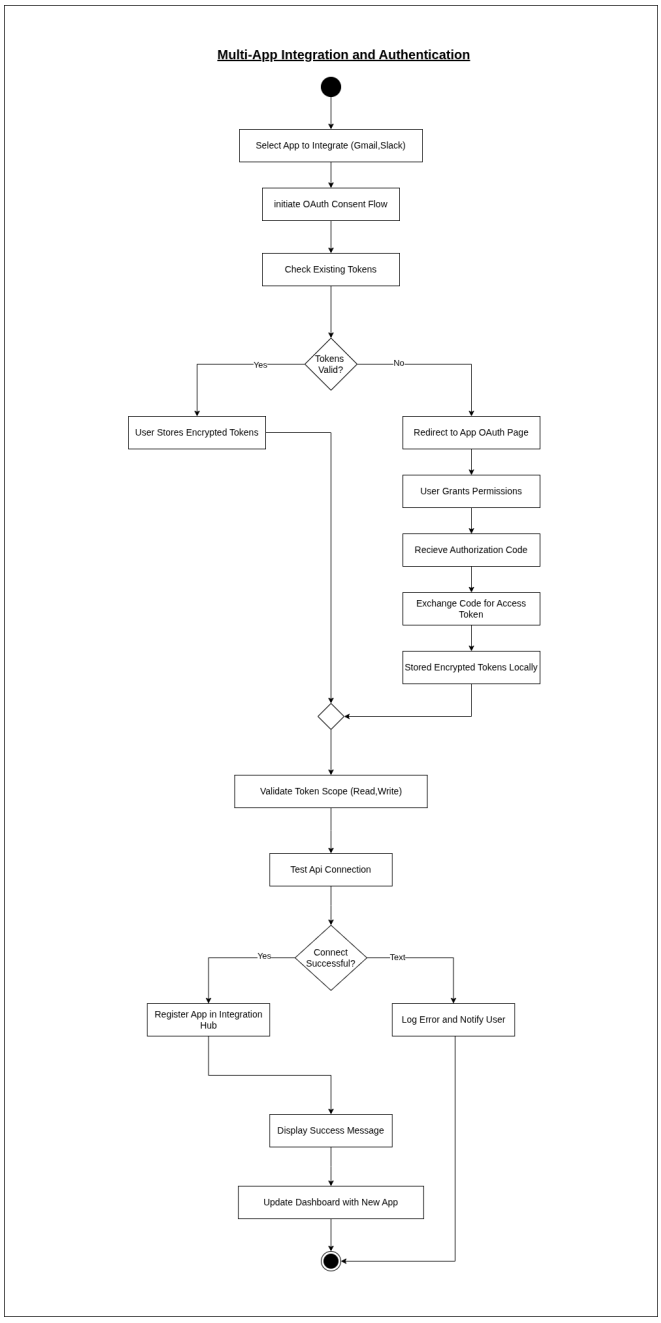

Figure 2.5: Activity Diagram – Multi-App Integration and Authentication

2.7 Component Diagram

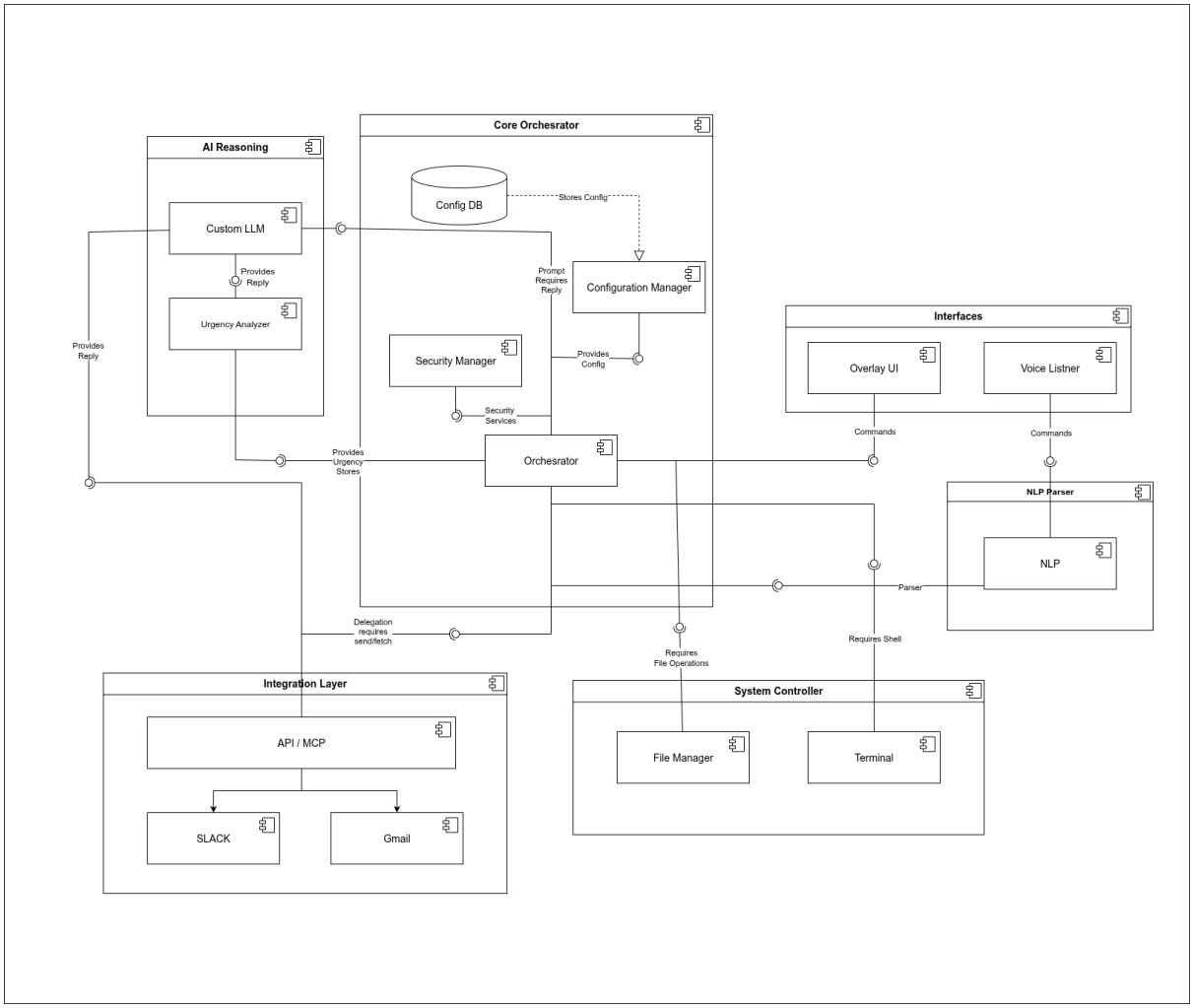

Figure 2.6: Component Diagram of Auto Return System

Chapter 3

Implementation

3.1 Development Environment

- **Language:** Python 3.10+
- **Core Framework:** FORGE (Python-native, async-first agent orchestration)
- **UI Framework:** Pyside6
- **AI/ML:** Ollama with Qwen3:0.6B
- **Database:** SQLite with vector extensions and Firebase
- **Async:** asyncio
- **Testing:** pytest
- **OS:** Linux

3.2 Current Implementation Status (as of Dec 2025)

Feature	Status
Priority of an email/message	**Algorithm Ready**
Email/Message Summary	**Implemented**
Draft Generation	**Implemented**
Plain Text Response	**Implemented**
Task Classification	**Implemented**
Event Driven File Lookup	**Pipeline Ready**
Intelligent Search Pipeline	**50% Done**
Desktop Notifications	**70% Done**
Sender Statistics	**60% Done**
Sentiment Analysis	**40% Done**
Pop-Up Notifications	**30% Done**
Missed Message Summary	**Pending (FYP-II)**
Voice Enabled Actions	**Pending (FYP-II)**
Meeting Preparation	**Pending (FYP-II)**
Adaptive Learning	**Pending (FYP-II)**

Table 3.1: Current Implementation Status of Features

3.3 Technical Challenges and Solutions

No.	Issue Name	Issue	Solution
1	No Dataset for Classification	No dataset available for classifying email topics and tasks. Couldn't fine-tune a model.	Researched and found **Qwen 3B** intelligent enough for reasoning and classification. Decided to use Qwen directly.
2	Gmail Automation Issues	Gmail API required complex OAuth setup. Token kept expiring.	Implemented clean OAuth workflow with **token auto-refresh** mechanism.
3	Priority Algorithm Not Suitable	Existing algorithms did not fit our email workflow. Could not reflect urgency, sender importance, or preferences.	Built **custom priority algorithm** from scratch including urgency, deadlines, sender weight, and message type.
4	No Keyword Dictionary Dataset	No dataset existed for priority-based keyword weights. No source for mapping keywords and their priority values.	Created default keyword dictionary based on research. Added **adaptive learning** (system updates weights over time based on user approval/preferences).
5	LLM Missing Context in Long Emails	Long emails and nested threads confused the model. Important info was sometimes ignored.	Added a **context extraction step** to keep only the relevant parts.
6	Intelligent Search Query Ambiguity	Natural-language queries were unclear (e.g., "hasnain meeting last 2 days"). Missing sender names or exact dates.	Planned to add **LLM-based query interpretation**. System converts query into sender + context + time range filters.

Table 3.2: Technical Challenges and Solutions

3.4 Key Technical Components

3.4.1 Voice Processing Pipeline

- Wake word detection ("Hey Auto")
- Speech-to-Text / Text-to-Speech
- Local processing for privacy

3.4.2 Agent Ecosystem

- Base agent framework
- Gmail integration
- Slack integration
- Jira integration
- Plugin system for extensibility

3.4.3 Intelligence Layer

- Intent classification
- Context management
- Task extraction
- Preference learning

Chapter 4

Testing and Evaluation

4.1 Test Cases – Part 1

ID	Scenario	Objective	Pre-conditions	Test Steps	Test Data	Expected Result
TC_INT_01	Verify multi-app integration (Slack + Gmail)	Ensure APIs connect successfully	Valid tokens	Connect apps and fetch messages	OAuth keys	Unified inbox retrieved
TC_AUTO_01	Auto email summarization	Verify NLP summaries	Gmail linked	Run summarization	Sample emails	Concise summaries shown
TC_TASK_01	Task extraction	Ensure tasks identified	Slack + Gmail active	Run task extractor	Message text	Prioritized task list generated
TC_NOTIF_01	Configurable alerts	Validate custom notifications	Event rule set	Trigger alert condition	"Interview call" email	Notification triggered
TC_DASH_01	Unified inbox display	Verify dashboard merges notifications	Apps linked	Open dashboard	Message events	Combined view accurate
TC_STATUS_01	Slack status voice control	Check voice-based status update	Mic active	Say "Mute Slack 1h"	Voice input	Slack muted for set time

Table 4.1: Test Cases – Integration and Automation

4.2 Test Cases – Part 2

ID	Scenario	Objective	Pre-conditions	Test Steps	Test Data	Expected Result
TC_REMIND_01	Follow-up reminder	Ensure reminders trigger correctly	Gmail + Slack active	Set reminder, simulate delay	Reminder = 3h	Alert generated
TC_SENTI_01	Sentiment analysis	Validate emotion/urgency detection	NLP model active	Analyze strong message tone	"URGENT! Project failed"	Marked as urgent
TC_LOG_01	Audit trail logging	Ensure action logs recorded	Admin access	Perform actions, open log	Log entries	Accurate timestamps
TC_ACCESS_01	Access revocation	Check permission removal	Active session	Revoke Gmail access	Gmail token	Access denied message
TC_OFFLINE_01	Offline functionality	Validate limited offline support	Cached data available	Disconnect, run summary	Cached threads	Summary shown locally
TC_SECURITY_01	Token encryption	Verify secure local storage	Tokens issued	Inspect encrypted store	Access tokens	Encrypted securely

Table 4.2: Test Cases – Notifications, Security, and Reliability

4.3 Testing Outcomes

All implemented modules passed functional testing. The priority classification algorithm was validated with test emails showing accurate urgency scoring. The LLM pipelines (summary, draft generation, task extraction) demonstrated acceptable accuracy and response times.

4.4 Key Technical Challenges Identified

1. **Local LLM Optimization:** Model size vs. performance trade-offs, response time optimization, context window management.

2. **System Integration:** Real-time sync across services, conflict resolution, error handling and recovery.

3. **Privacy & Security:** Local data processing, secure credential management, data encryp-

tion.

Chapter 5

Project Management (CLO4)

5.1 Project Timeline

Phase	Duration	Status	Semester
Phase 1: Research & Requirement Analysis	0–3 Weeks	Completed	Semester 1
Phase 2: System Design & Architecture	3–6 Weeks	Completed	Semester 1
Phase 3: API Research & Integration Testing	6–12 Weeks	Completed	Semester 1
Phase 4: Backend & Core System Development	12–18 Weeks	In Progress	Semester 1,2
Phase 5: Desktop App (Frontend)	18–22 Weeks	In Progress	Semester 1,2
Phase 6: Features Implementation	22–23 Weeks	In Progress	Semester 1,2
Phase 7: Smart Automation	23–26 Weeks	Pending	Semester 2
Phase 8: Testing, Evaluation & Finalization	26–28 Weeks	Pending	Semester 2

Table 5.1: Project Timeline

5.2 Risk Management

- **R1:** API token expiry → Mitigation: Implemented auto-refresh mechanism with secure local storage.
- **R2:** Voice recognition errors → Mitigation: Adaptive confidence thresholds and fallback to text input.
- **R3:** Integration failures → Mitigation: Retry mechanisms with exponential backoff and comprehensive logging.

- **R4:** LLM response inconsistency → Mitigation: Prompt engineering and response validation layers.
- **R5:** Cross-platform compatibility issues → Mitigation: Using PyQt6 for consistent native UI across platforms.

5.3 Team Work Distribution

- **Hasnain Saleem:** Core framework, LLM integration, priority algorithm
- **Alishba Tariq:** UI/UX development, testing, documentation
- **Kashan Saeed:** API integrations, database design, voice processing

Chapter 6

Investigation & SWOT

6.1 Research Analysis Overview

The project investigates existing solutions in unified communication platforms, AI-powered assistants, and productivity tools. Key investigation criteria include: Impact, Core Features, AI & Automation capabilities, Voice & NLP integration, Business Model, and Market Dynamics.

6.2 SWOT Analysis

6.2.1 Strengths

- Complete feature coverage across all categories with 20+ planned features
- Unique OS-level integration capabilities as a native desktop application
- Local-first AI processing for enhanced privacy and data security
- Voice-first design with custom wake words and natural language interaction
- Built on modular FORGE Framework for extensibility
- Cross-platform support (Linux, Windows, macOS)

6.2.2 Weaknesses

- Pre-revenue with unproven business model
- Small development team compared to established competitors
- No existing customer base or market validation
- Dependency on local LLM performance and hardware requirements

6.2.3 Opportunities

- Growing remote work market increasing demand for unified productivity tools
- Enterprise demand for privacy-focused AI solutions
- Partnership opportunities with SaaS providers and productivity tool vendors
- Open-source community contributions for plugin development
- Integration with emerging platforms and AI models

6.2.4 Threats

- Big tech companies replicating features in their existing products
- Privacy regulations impacting data processing requirements
- Rapidly evolving AI landscape making current implementations obsolete
- User resistance to changing established workflows
- Competition from both open-source and commercial alternatives

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The **Auto Return** system successfully demonstrates how intelligent automation and multi-agent coordination can unify productivity platforms into a seamless, privacy-first workspace. With the FORGE Framework as its foundation, the project has achieved significant milestones including:

- Successful implementation of core framework with 16+ Python modules
- Integration of Gmail, Slack, and Jira through agent-based architecture
- Development of custom priority classification algorithm
- Implementation of multiple intelligent pipelines (summary, draft generation, task extraction)
- Local LLM integration with Ollama and Llama 3.1 8B
- Cross-platform native desktop application using PyQt6

The system significantly reduces manual workload, minimizes context switching, and enhances productivity through intelligent automation while maintaining local-first privacy principles.

7.2 Future Work

1. **Complete UI/UX Implementation:** Finalize desktop interface and voice interaction system
2. **Advanced Features:** Implement pending features including voice control, custom wake words, meeting preparation, and adaptive learning
3. **Enhanced LLM Capabilities:** Integrate more advanced local LLMs and improve context management
4. **Additional Integrations:** Extend platform support to include Microsoft Teams, Discord, Trello, etc.
5. **Performance Optimization:** Improve response times and reduce resource consumption
6. **Comprehensive Testing:** Complete unit tests, integration tests, and end-to-end testing
7. **Deployment Packaging:** Create installation packages for different platforms with update system
8. **Community Development:** Foster open-source community for plugin and extension development

The project is well-structured with clear documentation and follows modern software engineering practices. It's designed to be extensible and maintainable, with a strong focus on AI agent integration and local-first privacy, positioning it as a promising solution in the productivity automation landscape.

References

Project Repository: https://github.com/hasnainsaleem18/WorkEase.git

Primary Reference Sources:

- Ferdium App – GitHub Repository: github.com/ferdium/ferdium-app
- Ferdi: A Free & Open-Source Alternative to Franz & Rambox – (It's FOSS)
- Rambox – Crunchbase: crunchbase.com/organization/rambox
- Wavebox – Official Website: wavebox.io
- Beeper – CB Insights: cbinsights.com/company/beeper-1
- Shift – Capterra: capterra.ca/software/169851/shift
- Superhuman – Sacra: sacra.com/c/superhuman/
- Front, Hiver, Missive – Shared Inbox Tools (HelpScout, CanaryMail)
- Respond.io – Wikipedia: en.wikipedia.org/wiki/Respond.io
- Chatwoot – Official Website: chatwoot.com
- AlternativeTo – Ferdi Alternatives: alternativeto.net/software/ferdi/

Technical References:

- Ollama Documentation: ollama.com

- PyQt6 Documentation: riverbankcomputing.com/software/pyqt
- Llama 3.1 Technical Paper: Meta AI Research
- SQLite with Vector Extensions: github.com/asg017/sqlite-vec
- FORGE Framework Architecture: Project Documentation

Funding & Market References:

- Respond.io Funding – TechNode Global: technode.global
- Respond.io Series A – Medium: medium.com/headlineasia
- Chatwoot – VentureBeat: venturebeat.com

Appendix A

Appendix

A.1 Project Repository

The complete source code, documentation, and design assets for the project are hosted on GitHub:

https://github.com/hasnainsaleem18/WorkEase.git

A.2 Research Review Table

No.	Author(s) & Year	Title / Source	Core Focus / Relevance to FYP
1	Bentahar, J. (2005)	*A Pragmatic and Semantic Unified Framework for Agent Communication*, Université Laval (Doctoral Thesis).	Develops unified framework for pragmatic and semantic aspects of multi-agent communication. Provides theoretical foundation for agent coordination and semantic understanding in automated workflows.
2	Cao, J. (2024)	*Deploying Large Language Models as Agents*, MIT CSAIL.	Connects LLMs to external APIs through autonomous agents. Blueprint for local LLM agent layer and orchestration.
3	Kumar, G. V., Penchala Jayanthi (2015)	*Real-Time Text and Speech Recognition System*, JNAO Journal, 2015.	Implements real-time speech-to-text and text-to-speech system using Google TTS and Python for natural language interfaces. Forms foundation for voice control and NLP integration in Auto Return.
4	Meta AI (2024)	*Llama 3.1 Model Technical Paper*, Meta AI Research.	Details the architecture and capabilities of Llama 3.1 models used for local LLM integration in our system.
5	FORGE Framework Team (2025)	*FORGE Framework Documentation*, Project Internal Documentation.	Provides the architectural foundation and implementation guidelines for our agent-based orchestration system.

Table A.1: Research Review Summary

A.3 Project Status Summary

- **Requirements & Design:** Completed with 25 comprehensive requirements and 17 detailed component designs

- **Core Framework:** 16+ Python modules implemented with event bus, LLM integration, and storage systems
- **Agent System:** Base agent framework with Gmail, Slack, and Jira integrations
- **Documentation:** Complete API documentation and developer guides
- **Current Focus:** UI/UX development, voice interaction system, notification system
- **Next Steps:** Complete testing, packaging for different platforms, and final documentation